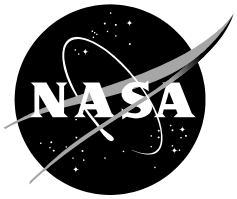
NASA/CR—2010–NNL07AA28C

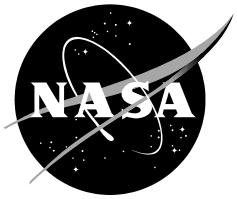


A Generalized Framework for Constrained Design Optimization of General Supersonic Configurations Using Adjoint Based Sensitivity Derivatives

Steve L. Karman Jr.
University of Tennessee at Chattanooga
Langley Research Center, Hampton, Virginia

February 2010

NASA/CR—2010–NNL07AA28C



A Generalized Framework for Constrained Design Optimization of General Supersonic Configurations Using Adjoint Based Sensitivity Derivatives

Steve L. Karman Jr.
University of Tennessee at Chattanooga
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, VA 23681

February 2010

Organization of Report

The first three section of this final report follow the organization of the initial proposal. Section 1 will reiterate the background material, describing the basic premise of the NASA NRA and the proposed research conducted during the execution of the contract. Section 2 will describe the general scope of the work performed. Section 3 describes each specific task of the Statement Of Work. A response to each task is included, identifying the outcome of the research in response to each task. Section 4 then describes the items that were delivered during the contract. Some of the items were specifically called out in the signed contract. Other items were supplemental items in the final data package that provide a complete suite of programs for performing meshing related tasks associated with mesh generation and adaptation for viscous supersonic flows. As each delivery item is described a sample input deck is included to help explain the input parameters of the programs. Section 5 uses a supersonic configuration to demonstrate the application of these meshing tools to two types of meshes. Finally, conclusions are presented summarize the outcome of the research.

1.0 Background

The Aeronautics Research Mission Directorate (ARMD) sent out an NASA Research Announcement (NRA) for proposals soliciting research and technical development. The proposed research program was aimed at addressing the desired milestones and outcomes of ROA (ROA-2006) Subtopic A.4.1.1 Advanced Computational Methods. The second milestone, SUP.1.06.02 Robust, validated mesh adaptation and error quantification for near field Computational Fluid Dynamics (CFD), was addressed by the proposed research. Additional research utilizing the direct links to geometry through a CAD interface enabled by this work will allow for geometric constraints to be applied and address the final milestone, SUP2.07.06 Constrained low-drag supersonic aerodynamic design capability.

The original product of the proposed research program was an integrated system of tools that can be used for the mesh mechanics required for rapid high fidelity analysis and for design of supersonic cruise vehicles. These Euler and Navier-Stokes volume grid manipulation tools were proposed to efficiently use parallel processing. The mesh adaptation provides a systematic approach for achieving demonstrated levels of accuracy in the solutions. NASA chose to fund only the mesh generation/adaptation portion of the proposal. So this report describes the completion of the proposed tasks for mesh creation, manipulation and adaptation as it pertains to sonic boom prediction of supersonic configurations.

2.0 Scope of Work

The contractor shall design, develop, and test an integrated framework for unstructured mesh generation and adaptation with direct links to geometry through solid modeler geometry kernels. The system shall produce valid purely tetrahedral and multi-element unstructured meshes that satisfy a prescribed anisotropic element sizing function are produced about arbitrary geometries without user intervention that are suitable for viscous analysis with sub-layer resolution for high Reynold's number flows. The system shall enable adaptive mesh refinement on meshes that produces elements that satisfy a prescribed anisotropic element sizing function. The integrated framework shall execute

efficiently on a cluster of commodity distributed memory computers.

3.0 Description of Tasks

This section describes the division of tasks as outlined in the original contract. Each subsection will begin with the text from the Statement of Work, followed by a response detailing how the task was completed.

3.1 Hierarchical Inviscid Mesh Generator

SOW:

The contractor shall rewrite the hierarchical inviscid mesh generator described in AIAA Paper 2004-0613 to operate on distributed memory parallel processors and incorporate general cutting of elements intersected by the geometry.

Response:

The parallel version of the inviscid mesh generation code P_HUGG was completed and delivered in executable form on October 10, 2007. The method was presented at an AIAA conference in January 2008.[1] A parallel mesh generation process was created to allow for rapid, automatic generation of inviscid meshes for arbitrary, complex geometry. This procedure uses Octree refinement of a Cartesian mesh with general cutting to create hybrid unstructured meshes. The meshing is completed in parallel using OpenMPI as the message passing protocol. Partitioning was originally performed using the Octree data structure or the METIS partitioning library, depending on the element/processor ratio. Multiple cases were run to test the robustness of the algorithm, including surface ships, airplanes, axial compressors and geographic landscapes. Parallel performance is documented in all 3D cases from 1 to 244 processors.

Subsequent developments led to a new partitioning scheme based on a split tree approach that provides better load balancing when an adaptation field is used to drive the mesh refinement. The latest code uses the SPACING FIELD library developed under this contract to control the refinement process when an adaptation spacing field is available. An example adaptive case is included in section 5 of this report. An updated executable is included in the final data package.

3.2 Interface to Multiple Solid Modeling Kernels

SOW:

The contractor shall develop an interface to multiple solid model geometry kernels through CAPRI (Robert Haimes, Massachusetts Institute of Technology) to maintain surface fidelity during initial grid generation and adaptation. The interface shall make use of a heterogeneous computing environment.

Response:

The CAPRI software library is being used by a Ph.D. student in research to develop a geometry preparation and design optimization console. The CAPRI interface was originally used with the ProE CAD kernel on Linux servers in a client/server mode. Recently the transition to the Solidworks CAD kernel running on a Window platform was completed. The console software is being developed on a Apple Mac Book Pro computer accessing CAPRI on the Window server across the network. In addition to the interactive console program, access to the CAD kernels will eventually be achieved through direct C/C++ calls from within the meshing programs P_HUGG, P_OPT, P_VLI and others. No deliverable is required for this CAPRI interface.

3.3 Prescribed Element Sizing

SOW:

The contractor shall use a prescribed element sizing to perform mesh adaptation. The adaptation shall be performed by refining the existing multi-element mesh. Auxiliary data associated with the mesh (i.e., flow solution) shall be transferred to the new mesh.

Response:

The ability to refine a multi-element mesh was initially completed and delivered in a serial version on April 10, 2009. A parallel version of this codes, P_REFINE, is included in the final data package. The prescribed element sizing can be generated using the SPACING program included in the final data package. SPACING reads a mesh and solution file from a Fieldvew plot file. Users can select one or more adaptation functions, such as pressure and Mach number. SPACING computes the gradient field of the selected function and determines a sizing field, which is stored in an HDF5 file. The refinement code P_REFINE reads that file and stores the sizing information in an OCTREE storage system (the OCTREE library is included in the data package) through the SPACING FIELD library. P_REFINE queries the sizing field through the SPACING FIELD library to determine the desired lengths of edges contained in the meshing being refined. Edges longer than the prescribed size are refined. A new Fieldview plot file containing the refined mesh and the transferred solution can be obtained using the INTERP code (included in final data package). INTERP makes use of the OCTREE library to store the solution variables from the initial Fieldview file and performs a simple interpolation of the variables to the new mesh. An example session of SPACING is included in Section 4.5. An example P_REFINE session is included in Section 4.2. An example session of an INTERP run is included in Section 4.7.

3.4 Unstructured Winslow Smoothing

SOW:

The contractor shall develop appropriate forcing functions to be used with an unstructured Winslow smoothing algorithm to control mesh spacing in regions with high gradients, such as shocks and boundary layers. These forcing functions shall maintain adequate normal spacing near viscous boundaries and maintain high quality elements throughout the domain. The contractor shall perform 2 to 5 test cases to determine the effectiveness of the forcing functions for specific types of meshes and flow field characteristics.

Response:

Research into unstructured Winslow smoothing has proceeded steadily during the execution of this contract. Intermediate results of the research were presented in an AIAA paper.[2] Forcing functions for controlling grid spacing have been investigated for an unstructured elliptic smoothing scheme. The forcing functions can provide control of grid spacing for feature-based solution adaptation and normal grid spacing for viscous clustering. An alternate approach, using Riemannian metric tensors to define a spacing field, was also described. This approach solves the same Winslow equations, but without forcing functions. Instead, the Riemannian metrics were used to alter the spacing of the computational mesh, which directly affects the spacing of the physical mesh. Two- and three-dimensional results were included to illustrate the use of these two different techniques. The modified computational control volume approach is more intuitive since the direct manipulation of the control volume in the computational domain is quantifiable and can be verified through visual inspection. Whereas the magnitude and size of the forcing function approach is less intuitive. Additional research was performed by a Ph.D.

student and was published in his dissertation and another AIAA presentation.[3,4] This latest approach used Virtual Control Volumes (VCV) to define the computational space. These idealized control volumes produced higher quality meshes in the physical domain and the method was more robust when performing large mesh deformations. Continued research into the construction of these VCVs in 3D continues. No deliverable was required for the unstructured Winslow smoothing.

3.5 Parallel Mesh Refinement

SOW:

The contractor shall incorporate distributed memory parallel processing into the mesh refinement capability. The existing grid shall be partitioned using standard parallel partitioning libraries. Several test cases shall be performed by the contractor to refine meshes with and without parallel processing to verify the parallel implementation.

Response:

The serial version of the multi-element refinement code delivered in April 2009 was modified to use MPI protocols to run in a parallel computer environment. One code (P_REFINE) can be executed in serial or parallel. The parallel execution can take place on a single shared memory computer with multiple processor or on a cluster of distributed memory computers. The input to run in serial or parallel is identical. An example parallel refinement session is included in Section 4.2. The source files and Makefile configuration for the parallel mesh refinement code is included in the final data package.

3.6 Parallel Viscous Layer Insertion

SOW:

The contractor shall incorporate distributed memory parallel processing into the viscous layer insertion method. The existing grid shall be partitioned using the standard partitioning libraries. The contractor shall modify the existing viscous insertion method to employ distributed memory parallel processing to insert viscous layers into an existing mesh. In addition, the contractor shall include the ability to introduce multiple normal vectors at sharp geometric edges, such as wing trailing edges

Response:

A parallel version of the viscous layer insertion code (P_VLI) was delivered in executable form on July 10, 2008. The code uses either the MeTis library or a split-tree based portioning scheme to partition the input mesh. The same code can be executed in serial or parallel, with the same input deck. MPI protocols are used to exchange data between processor. Multi-normal support was not incorporated after a decision was made to pursue the development of polyhedral support in future research. An example session is included in Section 4.3. P_VLI was also used in the cases included in Section 5. An updated executable is included in the final data package.

1.0 Description of Delivered Items

This section described the specific items delivered in the execution of the contract. Some items were specifically called out in the SOW. Other items are included to provide a complete suite of codes for the creation/manipulate/adaptation of meshes for supersonic flows. P_HUGG, P_OPT and P_VLI are delivered in executable form with restricted use. P_REFINE is delivered in source form with Makefiles for compiling on a Linux system. Some of the supplemental codes are delivered in executable for since they make use of the underlying data structures and routines of the restricted codes

(P_HUGG, P_OPT and P_VLI). Other supplemental codes are delivered in source form with associated Makefiles. A sample session where each program is executed is included in each subsection. These sample sessions will be included in the final data package. The order of the first three subsections follow the SOW, although the reader may find it more useful to read 4.1 followed by 4.4 and then 4.3 to understand the meshing process for the P_HUGG style meshes. P_REFINE (section 4.2) is usually performed after execution of SPACING (section 4.5).

4.1 P_HUGG (Task 3.1)

Geometry is provided to P_HUGG in the form of a triangulated surface mesh. This representation can be a Cart3D .tri file or an ACAD Facet file.[5,6] The Cart3D file format can be found on the Cart3D website. The ACAD file can be obtained from ACAD or it can be generated by Pointwise/Gridgen.[7] The geometry used in this final report is a wind tunnel test configuration that was used to model pressure signatures.[8] The surface triangulation for the aircraft and the other boundaries of the computational domain used in this report was created within Pointwise and exported as a Fieldview CRUNCH file and converted to a .tri file using the Conv program. The last entries in the .tri file are part numbers, which are used to group triangles for a given part, such as the body or wing, etc. The geometry file is included in the final data package. The geometry is shown in Figures 1, 2 and 3.

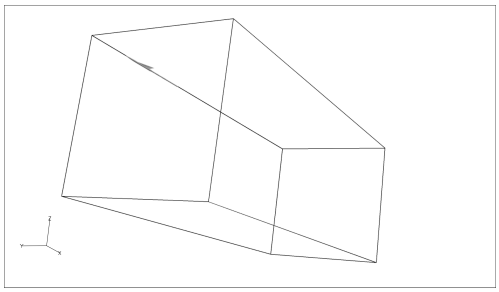


Figure 1. Bounding box for computational domain.

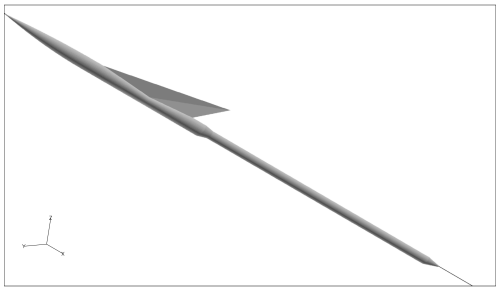


Figure 2. Supersonic model with sting.

Figure 3. Geometry surface triangulation.

When P_HUGG is executed interactively the user provides input on the command line. A journal file of those user inputs is created each time P_HUGG is executed. That journal file can be renamed, edited and provided to P_HUGG as input in subsequent sessions. The initial journal file is shown below:

```
#Number of geometry files >
1
#File 1 >
dw4_two_sym.tri
#Adaption spacing file present? [0=no, 1=yes] >
0
#Enter minimum spacing ->
0.05
#Enter maximum spacing ->
20
#Enter maximum X-Y aspect ratio ->
1
#Enter maximum X-Z aspect ratio ->
1
#Enter maximum Y-Z aspect ratio ->
1
#Enter cutting tolerance [ -ve multiplier, +ve absolute] ->
1e-10
#Enter boundary face merging tolerance [ degrees ] ->
20
#Enter boundary collinear point merging tolerance [ degrees ] ->
20
#Allow polygonal faces? [0=no, 1=yes] ->
0
#Allow polyhedra? [0=no, 1=yes] ->
0
#Allow surface quads? [0=no, 1=yes] ->
0
#Enter spawn level ->
8
```

Lines preceded by # are comment lines. These are the questions posed by P_HUGG. The user can input one or more geometry files. For this example only one is specified, dw4_two_sym.tri. When performing an adaptation using a spacing field created by SPACING the user will respond to the next question with a 1 indicating a spacing file is present and this will be followed by the file name for the spacing file. In the current example no spacing field data is specified. The user then inputs the global minimum and maximum spacing values. No Cartesian cubes smaller or larger than these limits will be permitted. P_HUGG can permit the generation of Cartesian elements with aspect ratios higher than one. This can be controlled in the three different Cartesian planes, X-Y, X-Z and Y-Z. The current version of P_HUGG is less robust when executed with values greater than 1, so unit aspect ratios are recommended. The cutting tolerance is used by the code to identify duplicate nodes when P_HUGG performs a general cutting of the Cartesian elements with the input geometry. A negative value will compute a tolerance based on the local smallest element size. A positive value is an absolute tolerance (RECOMMENDED). The cutting process will create a collection of polygons on the surface that are bounded by the Cartesian element extents and the original geometry triangles. Within each Cartesian element these polygons associated with given body part can be merged using the input boundary face merging tolerance angle. Any polygons for a given body within a Cartesian element that have a turning angle with a neighboring polygon in that same Cartesian element that is smaller than the tolerance angle are combined. The number of points in the bounding curves of those polygons can be reduced using the input boundary collinear point merging tolerance. Segments of the bounding curve that have a turning angle less than the tolerance are candidates for merging. Merging will not occur for nodes that are deemed critical, such as corner nodes of the element or nodes in the middle of the face that are corner nodes of adjacent elements. The next input determines whether P_HUGG will output the mesh with polygon faces (1) or triangular and quadrilateral faces (0). This is followed by a flag that determines whether the volume elements are polyhedra (1) or mixed element types (0). If mixed element is selected then any polyhedra are converted into tetrahedral, pyramid, prisms and hexahedra as necessary. This may introduce additional nodes at the center of the polyhedra. The next input determines whether surface quadrilateral are permitted when polygonal faces are not permitted. If set to zero then the surface polygons and quadrilaterals are triangulated. The final input is the hierarchical level in the meshing process where the ownership of the volume elements is transferred to the parallel processors. Before that level all processors are creating an identical mesh. At spawn the elements on the finest level are partitioned using the split-tree partitioning scheme described in the Conv program. The meshing proceeds in parallel after spawn.

If this journal file is renamed and edited by the user it can be specified as an input deck on the command line after the executable name (P_HUGG.LINUX64). Another journal file will be created during the run.

When the input geometry is read P_HUGG will look for a file called P_HUGG_geometry.params. This file contains the list of parts included in the geometry and some input parameters for each part that controls how P_HUGG and other programs treat each boundary. If the file does not exist P_HUGG creates it for the current geometry. The file created for the sample geometry is shown below.

```
Number of boundary files
1
dw4_two_sym.tri
Number of boundaries
14
Boundary #   Layers      g_space       n_space              boundary name
---------- ---------- -------------- -------------- --------------------------------
         0          0         -5e+18         -5e+18 Virtual
```

```
 1          0     0.01037598     0.01037598 Boundary_1
 2          0     0.02075195     0.02075195 Boundary_2
 3          0     0.01037598     0.01037598 Boundary_3
 4          0    0.001296997    0.001296997 Boundary_4
 5          0       1.328125       1.328125 Boundary_5
 6          0       1.328125       1.328125 Boundary_6
 7          0       1.328125       1.328125 Boundary_7
 8          0       1.328125       1.328125 Boundary_8
 9          0     0.02075195     0.02075195 Boundary_9
10          0     0.04150391     0.04150391 Boundary_10
11          0   8.106232e-05   8.106232e-05 Boundary_11
12          0     0.02075195     0.02075195 Boundary_12
13          0       1.328125       1.328125 Boundary_13
14          0       1.328125       1.328125 Boundary_14
```

This file may be edited to change the boundary names and parameters. Boundary 0 is a virtual boundary and can be used to control mesh spacing in the domain for geometry triangles that are tagged with part number 0 in the .tri file. This is a seldom-used capability and recommended for advanced users only. For inviscid meshing the g_space column specifies the default desired spacing for each geometry triangle of that boundary. Cartesian volume elements in the vicinity of each geometry triangle are tested against this size parameter. Elements with a dimension larger than this size are refined. The Layers column specifies whether viscous layers will be inserted with the initial spacing of n_space when P_VLI is executed. When P_HUGG is executed the Layers flag can be used to trigger some adjustments to the inviscid spacing at the surface triangles. An adjustment based on approximate curvature is performed when the Layers flag is set to 1. If the Layers flag is set to 2 then P_HUGG will perform an intersection/thinness test for each surface geometry triangle and it will use the n_space parameter as the initial desired inviscid normal spacing. If the Layers flag is set to 3 then both curvature and intersection/thinness adjustments are performed. A modified P_HUGG_geometry.params file with the boundaries named and some changes to the Layers flag and g_space is shown below.

```
Number of boundary files
1
dw4_two_sym.tri
Number of boundaries
14
Boundary #   Layers      g_space       n_space              boundary name
---------- ---------- -------------- -------------- --------------------------------
         0          0         -5e+18         -5e+18 Virtual
         1          1     0.05000000     0.01037598 Bevel
         2          1     0.10000000     0.02075195 Body_LL
         3          1     0.05000000     0.01037598 Cone
         4          1    0.001296997    0.001296997 Cone_Base
         5          0       5.000000       1.328125 Far_bottom
         6          0      10.000000       1.328125 Far_inflow
         7          0      10.000000       1.328125 Far_left
         8          0      10.000000       1.328125 Far_outflow
         9          1     0.10000000     0.02075195 LL1
        10          1     0.10000000     0.04150391 LL2
        11          1   8.106232e-05   8.106232e-05 Nose
        12          1     0.20000000     0.02075195 Sting
        13          0       1.328125       1.328125 Y_sym
        14          0       1.328125       1.328125 Z_sym
```

The resulting P_HUGG mesh for the journal file listed earlier and the parameter file listed above is shown in Figure 4.This run was executed on a Mac Book Pro using two processors. The outline of the two partitions is shown in the figure.

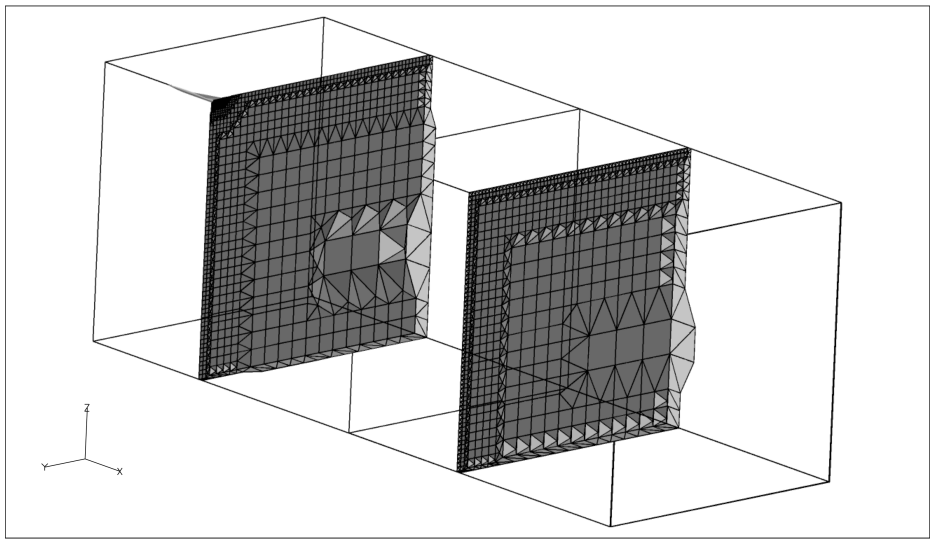


Figure 4. Cutting planes shown for X=15 and X=100.

Output from P_HUGG is written to a file for each processor. In this case two files are created P_HUGG_out.0 and P_HUGG_out.1. A CGNS files is also written by each processor, P_HUGG_0.cgns and P_HUGG_1.cgns. These files can be combined into a single file using Conv. They may then be converted of other formats, such as Fieldview.

The mesh made by P_HUGG may need to be optimized to produce valid surface and volume elements. The cutting process can create small slivers and inverted elements near the boundaries. P_OPT is used to smooth the mesh and provide a high quality inviscid mesh, ready for viscous layer insertion.

4.2 P_REFINE (Task 3.3 and Task 3.5)

Adaptive subdivision refinement of hybrid meshes is accomplished using P_REFINE. It can be executed in serial or parallel. When executed in parallel the input mesh files are the partitioned CGNS files created by running CONV using option 2 to partition a mesh. The adaptive spacing field is an HDF5 file created by running SPACING on a Fieldview plot file. To demonstrate the process the supersonic configuration detailed in Figures 1 and 2 was meshed in Pointwise, creating an all-tetrahedral mesh. Viscous layers were then inserted using the same process as described in Section 4.3 except the input mesh to P_VLI was the Pointwise tetrahedral mesh instead of the P_HUGG mesh. The resulting viscous mesh was analyzed using the SimCenter Navier-Stokes code Tenasi for Mach 1.68 flow at a Reynold's number of 4.13e06. Details of this analysis are included in Section 5. The Fieldview plot file from Tenasi was used with the SPACING code described in Section 4.5 to create a spacing field with pressure selected as the adaptation function. The resulting X component of the spacing field on the Y symmetry plane is shown in Figure 5. The color range is restricted between 0.01 and 5 do illustrate the variation in the sizing field. The actual range computed by SPACING ranged from 5e-04 to 17.

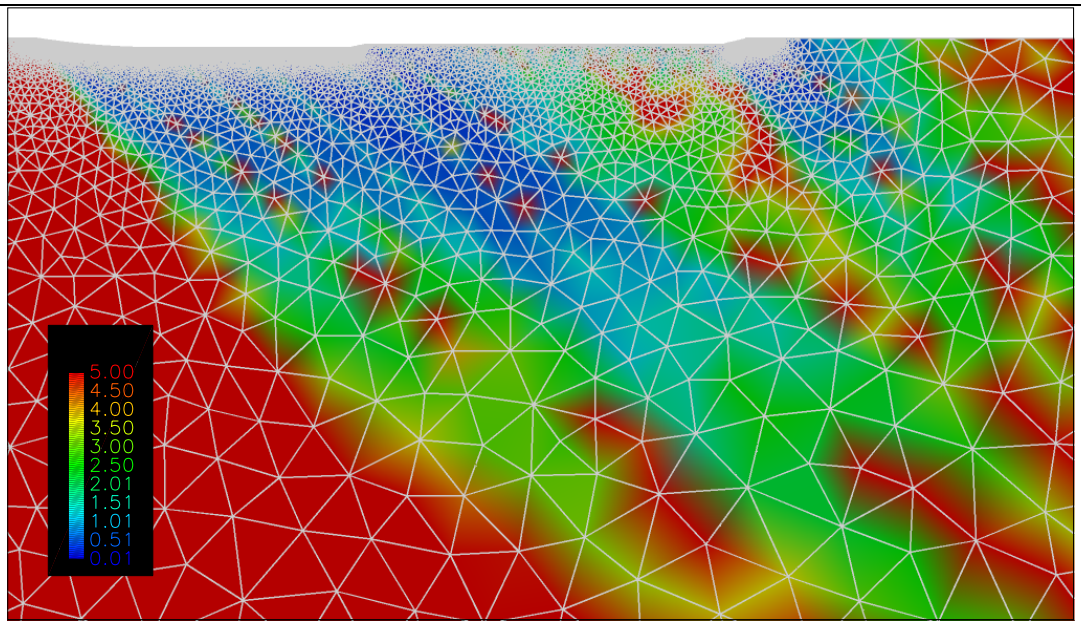


Figure 5. X component of pressure spacing field for initial solution.

This spacing field was input to P_REFINE with the original viscous mesh and refined. The journal file that was produced by P_REFINE is shown below.

```
#Adaption file name ->
adapt.hdf5
#Enter physical grid file name (serial version) ->
pw_refined.cgns
#Enter target number of new points ->
1000000
#Prohibit 2-node triangle marking [0 1] ->
1
#Force prism stack refinement [0 1] ->
1
#Enter metric length refinement threshold [ >= 1.0] ->
1.1
```

The spacing field was contained in file adapt.hdf5. The input mesh, pw_refined.cgns is the viscous mesh that was analyzed using Tenasi. This mesh file is overwritten by P_REFINE, so care should be taken to preserve a copy if needed. The user can restrict the amount of refinement using the target number of points parameter. This parameter defines the number of edges that will be marked for refinement based on the edge metric length computed from the spacing field. These metric lengths are ranked from highest to lowest. As indicated in the input, the first 1000000 edges will be tagged for refinement. The remaining edges will not be marked for refinement, but may get refined as the mesh quality constraints are imposed. If the target number is large then P_REFINE will be free to mark as many edges for refinement as dictated by the adaptation functions. The next input, prohibiting 2-node triangle marking, will force triangles to be refined producing only sub-triangles and tetrahedral will be forced to refine producing only sub-tetrahedra. The prisms in the viscous layers can be forced to refine consistently through the vertical stack of elements using the next parameter. This will prevent the subdivision of viscous prisms resulting in extremely large angles. The

final parameter is a metric length threshold that defines when edge refinement is imposed. The metric length for an edge, AB, is defined as

$$d_{AB} = \sqrt{\overrightarrow{AB}^T \quad [M] \quad \overrightarrow{AB}} \qquad (1)$$

where the matrix M is the Riemannian metric tensor, defined in detail in Section 4.5. A metric length of 1 means the edge is the proper length, according to the spacing field. A metric length greater than 1 means the edge is too long, so specifying a threshold of 1.1 will prevent refinement of edges that are within 10% of the proper length.

Figure 6 shows the original mesh on the Y symmetry plane. Figure 7 is the refined mesh using the spacing field. It is apparent in the figure that the elements below the vehicle have been refined.

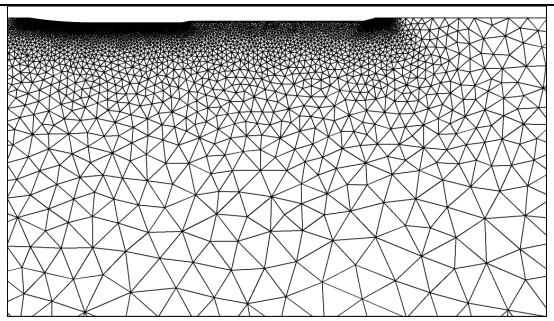


Figure 6. Original mesh on Y symmetry plane.

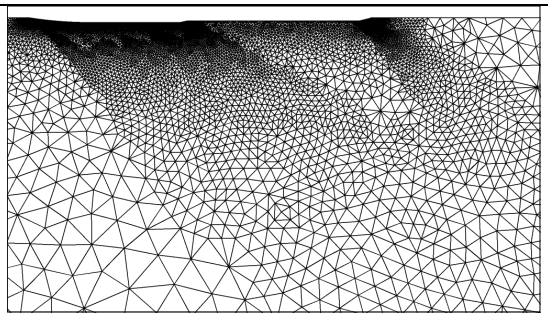


Figure 7. Refined mesh on Y symmetry plane.

Additional solution and refinement passes can be performed. A sequence of five refinement passes is detailed in Section 5 for this case. More details of the use of SPACING and P_REFINE can be found in an AIAA paper where the meshing tools developed under this contract were applied to a Navier-Stokes analysis of an F-35 in forward flight and hover.[9]

4.3 P_VLI (Task 3.6)

P_VLI uses linear-elastic smoothing to insert viscous layers into an existing unstructured mesh. The details of the method can be found in an AIAA Journal paper.[10] The basic premise is to push the existing mesh away from the surface, creating room for a layer of elements to be inserted at the boundary. If the surface elements are triangles then triangular prisms are inserted. If the surface elements are quadrilaterals then hexahedra are inserted. The scheme can proceed one layer at a time, inserting the top viscous layer first and the bottom layer last. Or the scheme can insert all layers at once. If the one layer at a time approach is used then P_VLI will measure the normal spacing of the existing mesh and begin inserting layers when sufficient room exists for that layer. This allows for a different number of elements in the stack of prisms/hexahedra that is dependent on the local mesh spacing. If the all layers at once approach is used then P_VLI will measure the size of normal spacing at the surface and attempt to insert a sufficient number of layers locally that allows for matching of the thickness of the top layer with the existing local mesh spacing. The layer-by-layer approach will produce more viscous layers, but it will take considerably longer time to complete and may encounter mesh tangling if too many layers are requested. The all at once approach will be much faster, is very robust, but will not necessarily match the top layer spacing with the local mesh and will result in a thinner viscous layer region. The all at once approach is demonstrated below.

The optimized mesh is the starting point for viscous insertion. P_VLI can work with P_HUGG style meshes or other meshes, such as the meshes created using Pointwise described in this report. The Layers parameter in the P_HUGG_geometry.params file is used to indicate which body parts require viscous layers. The initial normal spacing is defined in the n_space parameter in the file. The P_HUGG_geometry.params file modified for viscous layer insertion is shown below. The initial normal spacing is specified as 5e-05 for the viscous boundaries that are indicated with a Layers flag of 1.

```
Number of boundary files
1
dw4_two_sym.tri
Number of boundaries
14
Boundary #   Layers      g_space       n_space              boundary name
---------- ---------- -------------- -------------- --------------------------------
         0          0         -5e+18         -5e+18 Virtual
         1          1     0.05000000          5e-05 Bevel
         2          1     0.10000000          5e-05 Body_LL
         3          1     0.05000000          5e-05 Cone
         4          1    0.001296997          5e-05 Cone_Base
         5          0       5.000000       1.328125 Far_bottom
         6          0      10.000000       1.328125 Far_inflow
         7          0      10.000000       1.328125 Far_left
         8          0      10.000000       1.328125 Far_outflow
         9          1     0.10000000          5e-05 LL1
        10          1     0.10000000          5e-05 LL2
        11          1   8.106232e-05          5e-05 Nose
        12          1     0.20000000          5e-05 Sting
        13          0       1.328125       1.328125 Y_sym
        14          0       1.328125       1.328125 Z_sym
```

Input to P_VLI can be interactive or read from an input deck. As with the other meshing programs, the input deck can be a journal file from a previous session that was renamed and edited to suit the current viscous insertion process. The input deck for this demonstration of inserting viscous layers in the P_HUGG style mesh created in Section 4.1 and optimized in Section 4.4 is shown below.

```
#Number of geometry files ->
1
#File 1 ->
dw4_two_sym.tri
#Enter physical grid file name, serial version ->
dw4.cgns
#Enter number of viscous layers to insert >
-35
#Enter stack height ratio to current height [ >= 1.0] >
2.0
#Enter 1st pass # of outer smoothing sweeps >
1
#Enter 2nd pass # of outer smoothing sweeps >
8
#Enter # of Linear-Elastic smoothing sweeps (updated each time) >
50
#Enter stiffness model [0-AR, 1-AR/vol^2, 2-AR/vol^2/ds, 3-Unity] >
0
#Enter kappa [ >= 0.0] >
0
#Enter # of normal smoothing sweeps >
10
#Enter convergence OoM [ > 0] >
4
#Enter # of GMRES search directions [ > 0] >
30
#Enter initial geometric progression factor [1.0-1.5] >
1.15
#Enter geometric progression growth factor [1.0-1.5] >
1.01
#Enter minimum geometric progression factor [1.0-1.5] >
```

```
1.15
#Enter maximum geometric progression factor [1.0-1.5] >
1.25
#Overwrite restart at each layer [0 1] >
1
```

The geometry files are input first, exactly the same as required by P_HUGG and P_OPT. The serial version of the mesh file is entered next. If P_VLI is executed in parallel then this filename will be augmented by P_VLI to reflect the processor dependent file names, consistent with how Conv partitions the CGNS file. In other words, this input deck always reflects the input parameters as if the run is serial. The number of viscous layers is input next and for the all-at-once approach this number is negative. Two different outer smoothing sweeps are input next. This is mainly for the layer-by-layer approach. If inserting layer by layer then the initial number of outer smoothing passes is used by the linear elastic smoother, typically one outer pass. If that fails due to mesh buckling then the process for that layer is restarted with the second number of outer smoothing passes specified. When the second pass is needed the linear elastic smoother will divide the perturbation amount by the number of outer smoothing steps. This will allow the scheme to adjust the stiffness parameters as the mesh is deformed, which can prevent the element buckling that occurred in the first pass.

If the all-at-once approach is used then the second number of outer smoothing passes is used and the first outer smoothing pass parameter is skipped. When the all-at-once approach is employed the user is asked to enter the desired stack height as a multiplier on the current element height. The value of 2.0 input will allow the stack of prisms to be twice as tall as the current element locally. This enables the code to generate layers that better match the existing element size at the top layer of the stack. A value of 2 is conservative. This can be adjusted through trial and error for any configuration to achieve a better spacing match. If mesh buckling occurs then this parameter should be decreased. This stack height ratio parameter is not requested when using the layer-by-layer approach.

Next the number of linear-elastic smoothing passes is specified. This is the number of linear solve iterations prescribed. The method uses GMRES to solve the equations, but it is explicit at partition boundaries, so multiple iterations are required with a data exchange between processor after each iteration. It can also be thought of as a GMRES restart where the number of GMRES search directions can be controlled to manage the memory required by the method. Typically 50 iterations are sufficient.

The stiffness model is specified next. There are several models incorporated and each has its strengths and weaknesses, depending on the application. The model that has worked the best for viscous insertion is model 0, based on element aspect ratio. This allows the stiffness to increase for higher aspect ratio elements and help prevent further degradation of the element quality, but allows lower aspect ratio elements to deform as necessary to enable the mesh movement. The kappa parameter specified next is a new parameter that adds additional terms to the linear-elastic equations to enforce rigid body type rotations. It was introduced for mesh movement problems, not viscous insertion problems, so a value of zero is recommended.

P_VLI computes normal vectors for all the surface nodes where insertion is occurring. The number of sweeps specified will allow P_VLI to blend these normal vectors across the surface. Usually 10 or less normal smoothing passes are sufficient.

A convergence Order of Magnitude is specified next. Full convergence is not required for successful viscous insertion. Usually 3 to 4 orders are sufficient. This is followed by the number of GMRES search directions. A large number is not recommended as this will considerably increase the memory requirements. Plus the partition interfaces are explicit, so a parallel run would not converge anyway with pure

GMRES iterations. Thirty search directions are usually sufficient. The scheme will reset with the current solution at the outer iteration level, so convergence of the entire system is still achieved.

Now the normal spacing distribution is specified using a geometric progression scheme. The initial progression factor is specified as 1.15, which means the second layer thickness is 15% taller than the first layer. This progression factor is multiplied by the growth rate of 1.01, which means the progression factor for the third layer is 1.1615 or 16.15% thicker than the second layer. A growth rate of 1.0 would maintain a constant progression factor. Using the growth rate allows for fewer layers in the viscous region. The absolute limits on the progression factor are specified next, which would bound the progression factor between the initial value and 1.25.

Finally, the option to output the mesh after the successful completion of each layer is input next. This has no bearing on the all-at-once approach. For the layer-by-layer approach the code will write CGNS files after each layer is inserted. This allows one to obtain a viscous mesh in the case where buckling occurred in the last few layers and the scheme was unable to completed. In that case the last successful layer inserted would not be at the prescribed initial spacing, but it is a valid mesh.

The viscous mesh produced using this input deck is shown in Figure 8. The prism layers are connected to pyramid elements produced by the general cutting. Those pyramid elements are then connected to hexahedral elements. The all-at-once approach did a good job of matching the spacing of the pyramid elements. The input deck requested 35 layers of prism. P_VLI was able to insert a minimum of 5 layers and a maximum of 25 layers. The final mesh contained 1,461,430 nodes, 632,326 tetrahedra, 542,932 pyramids, 1,963,490 prisms and 121,572 hexahedra. The total wall clock time for inserting layers on a MacBook Pro using 2 processors was 1 hour 22 minutes.

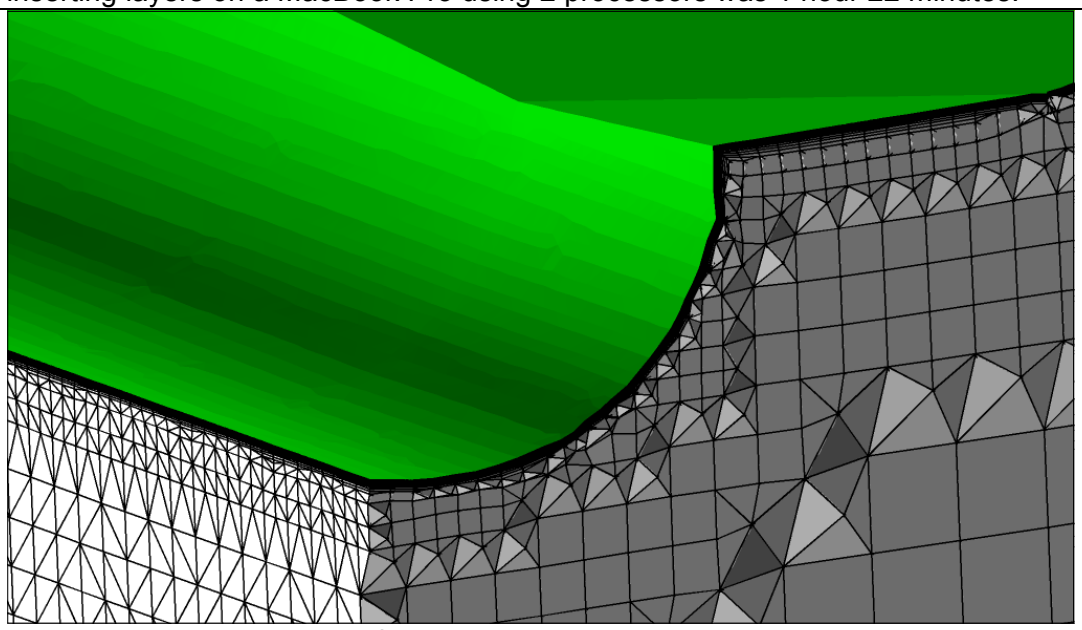


Figure 8. View of viscous layers inserted using input deck.

4.4 P_OPT

P_OPT is a general-purpose mesh-smoothing program that can be used to improve the quality of inviscid meshes. It can also perform some adaptation through mesh movement, provided a spacing field is available. This is a preliminary capability and computationally intensive, therefore it is not recommended. If P_OPT is applied to a

viscous mesh the quality of the viscous layers may be compromised, so that is also not recommended. The basic algorithm employed by P_OPT is a node perturbation scheme that seeks to reposition the node to minimize a cost function. The cost function for a node is based on the condition number of the elements surrounding the node. The method can detect when an element is inverted and is able to correct the inversion.

P_OPT requires a geometry definition, just as P_HUGG and P_VLI do. In fact, P_OPT will look for the same P_HUGG_geometry.params file in the working directory. If the file is not found P_OPT will create it. The Layers flag and the spacing parameters have no influence on the operation of P_OPT. The only important features of the parameter file are the geometry file name(s) and the boundary names.

As with the other meshing programs, a journal file is created when P_OPT is executed. The file can be renamed, edited and specified on the command line as an input deck on subsequent runs of P_OPT. The journal file used to smooth the sample mesh created by P_HUGG is shown below.

```
#Number of geometry files ->
1
#File 1 ->
dw4_two_sym.tri
#Adaption spacing file present? [0=no, 1=yes] >
0
#Enter physical grid file name, serial version ->
opt.cgns
#Enter # of smoothing sweeps >
50
#Enter boundary movement flag [0 1] >
1
#Enter interior movement flag [<0, 0, >0] >
4
#Enter threshold parameter [0.0-1.0] >
0.1
#Enter surface cost flag [0-no, 1-yes, 2-mixed] >
2
#Enter expanded stencil flag [0-no, 1-yes, 2-mixed] >
1
```

The physical grid file, opt.cgns, was a copy of the files produced by P_HUGG and reassembled into one file by Conv. Although not indicated by the journal file, this P_OPT run was executed in parallel using four Linux processors. Conv was used to partition the mesh file using the split-tree algorithm. Fifty smoothing iterations were specified. The boundary movement flag indicates that boundary nodes are allowed to move. This means the surface mesh points will be perturbed and re-projected onto the surface defined by the .tri file. The interior movement flag was specified as 4. This flag specifies how many layers of nodes normal to the surface are allowed to move. Each surface node is labeled layer zero. The next layer of nodes is layer one and so on. So a large number specified for the interior movement flag will allow all interior nodes to be optimized. With the movement flag was specified as 4 then just the first four layers of nodes are free to move. If the flag is specified as a negative, such as -4, then all but the first 4 layers are free to move. A large positive number allows all interior nodes to move.

The threshold parameter defines a cost function level for optimization. Nodes with a cost function above the threshold are optimized. Nodes with a cost function below the threshold are not optimized. This is a time saving measure since these optimization schemes can be computationally intensive. A low cost function is desired. A cost equal to one indicates a collapsed element and a cost greater than one means an element is inverted. Typical values for the threshold are 0.1 or lower. A cautionary note: this

optimization scheme will not fully converge. If the threshold is very small there will always be some perturbations performed on the mesh, a jittering so to speak. So don't expect to run P_OPT and achieve typical convergence. The maximum cost function in the range of 0.1 or lower is considered good.

The surface cost flag specifies how the cost function for the surface nodes is computed. Option 1 ignores the volume element and defines a cost based on 2D surface elements. Option 0 uses the volume cost function. Option 2 uses a mixed cost function. The surface cost function (option 1) will detect inverted triangles/quadrilaterals and move the surface nodes in a direction on the surface that minimizes the cost based on 2D elements, ignoring the effect of the motion on the volume elements. The option using volume cost function (option 0) will use the same definition of the cost as interior nodes. If mixed (option 2) was specified then the scheme will use the surface cost until the surface element inversion is eliminated and then revert to the volume cost function. Option 2 is the recommended approach.

The expanded stencil flag was an option added to test the effectiveness of the scheme using a smaller stencil for the cost function. The recommended value is 1 and should not be changed.

P_OPT will write the output to a single file, P_OPT.out, whether run in serial or parallel. The code will output the minimum, average and maximum cost function at each iteration. In addition, the number of nodes exceeding the threshold, the number of nodes with element inverted and the location of the maximum cost are printed. The maximum cost should be monitored closely. It should be reduced below 1 into the range of 0.1 or lower. Output from the sample run is included in the data package.

P_OPT will overwrite the CGNS file(s) after completing the run. If executed in parallel the multiple CGNS files can be recombined using Conv. The outcome from P_OPT is shown in Figures 9 and 10. The original mesh produced by P_HUGG is shown in Figure 9. The optimized mesh is shown in Figure 10. P_OPT tends to preserve the Cartesian character of the mesh, but eliminates the small sliver elements near the surface that can result from the general cutting process.

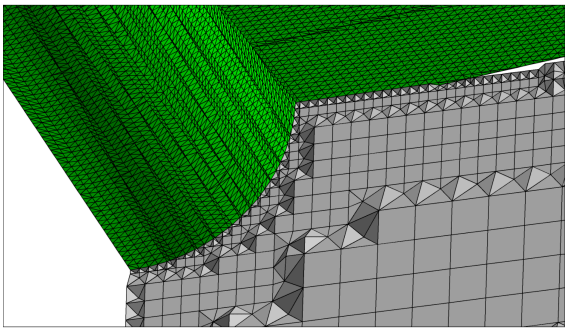


Figure 9. Surface mesh (green) and mesh at X=15 cutting plane (gray) before mesh optimization.

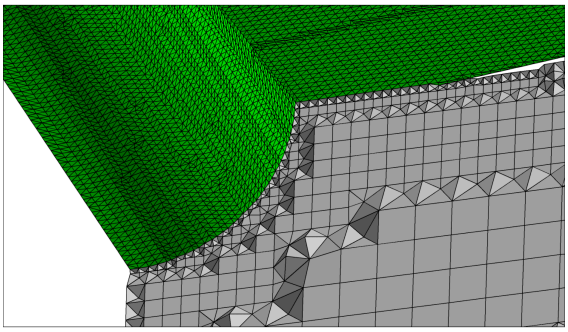


Figure 10. Mesh after performing optimization smoothing.

4.5 SPACING

The program SPACING, delivered as source, is used to create a spacing field for mesh adaptation/refinement that is based on identifying features in the solution indicated by high gradients of user selected solution parameters. This program reads a Fieldview

plot file that contains solution information. One or more solution functions are selected by the user. An adaptation function is computed for each edge in the Fieldview mesh based on the gradient vector of the selected function(s) and the edge vector. The statistical mean and standard deviation are determine for each adaptation function, which are then used to compute a size parameter to be store at the edge or at the nodes. Sizing information is then output to a binary (hdf5) file to be read by the SPACING FIELD library. The process of creating these size parameters will be described now in more detail, as a spacing field is created for a sample case.

SPACING is an interactive program, but can be run using an input deck that was created as a journal file from a previous session. The user inputs the name of the hdf5 file used to store the spacing field and the number of levels allowed in the Octree storage system. Thirty levels are usually sufficient.

```
MESH SPACING Program:
#Enter name of output spacing file >adapt.hdf5

#Enter maximum level in output Octree >30
```

Next, the name of the Fieldview file is specified, along with a scaling parameter used to scale the mesh to be consistent with the mesh to be adapted.

```
#Is there a Fieldview file? [0 1] >1

#Enter the name of Fieldview file >pw1.b4.fv

Filename = <pw1.b4.fv>
Fieldview version number 2.5
Fieldview file contains grids & solution!
Number of boundaries = 14
 Input boundary 1 name = Bevel
Stored boundary 1 name = Bevel
 Input boundary 2 name = Body_LL
Stored boundary 2 name = Body_LL
 Input boundary 3 name = Cone
Stored boundary 3 name = Cone
 Input boundary 4 name = Cone_base
Stored boundary 4 name = Cone_base
 Input boundary 5 name = Far_bottom
Stored boundary 5 name = Far_bottom
 Input boundary 6 name = Far_inflow
Stored boundary 6 name = Far_inflow
 Input boundary 7 name = Far_left
Stored boundary 7 name = Far_left
 Input boundary 8 name = Far_outflow
Stored boundary 8 name = Far_outflow
 Input boundary 9 name = LL1
Stored boundary 9 name = LL1
 Input boundary 10 name = LL2
Stored boundary 10 name = LL2
 Input boundary 11 name = Nose
Stored boundary 11 name = Nose
 Input boundary 12 name = Sting
Stored boundary 12 name = Sting
 Input boundary 13 name = Y_sym
Stored boundary 13 name = Y_sym
 Input boundary 14 name = Z_sym
Stored boundary 14 name = Z_sym
Number of variables = 10
Variable 1, name = Density
Variable 2, name = X_Momentum; Momentum
Variable 3, name = Y_Momentum
Variable 4, name = Z_Momentum
Variable 5, name = TotalEnergy
Variable 6, name = Temperature
Variable 7, name = Turb_KE
Variable 8, name = Distance_Function
Variable 9, name = Eddy_Viscosity
```

```
    Variable 10, name = Part
    Number of boundary variables = 5
    Boundary variable 1, name = HeatFlux
    Boundary variable 2, name = Cp
    Boundary variable 3, name = Cf
    Boundary variable 4, name = YPlus
    Boundary variable 5, name = wallY
    Reading grid 1
    Number of new nodes = 312613
    Body 1, # of faces= 252
    Body 2, # of faces= 8260
    Body 3, # of faces= 840
    Body 4, # of faces= 296
    Body 5, # of faces= 1966
    Body 6, # of faces= 912
    Body 7, # of faces= 1470
    Body 8, # of faces= 898
    Body 9, # of faces= 2248
    Body 10, # of faces= 1966
    Body 11, # of faces= 416
    Body 12, # of faces= 2268
    Body 13, # of faces= 17273
    Body 14, # of faces= 17103
    Body 1, # of faces= 0
    Body 2, # of faces= 0
    Body 3, # of faces= 0
    Body 4, # of faces= 0
    Body 5, # of faces= 0
    Body 6, # of faces= 0
    Body 7, # of faces= 0
    Body 8, # of faces= 0
    Body 9, # of faces= 0
    Body 10, # of faces= 0
    Body 11, # of faces= 0
    Body 12, # of faces= 0
    Body 13, # of faces= 10683
    Body 14, # of faces= 11474
     # of new tetrahedra = 526770
     # of new pyramid    = 550
     # of new prism      = 399413
     # of new hexahedra  = 0
    Body 1 has 252 triangles, 0 quadrilaterals.
    Body 2 has 8260 triangles, 0 quadrilaterals.
    Body 3 has 840 triangles, 0 quadrilaterals.
    Body 4 has 296 triangles, 0 quadrilaterals.
    Body 5 has 1966 triangles, 0 quadrilaterals.
    Body 6 has 912 triangles, 0 quadrilaterals.
    Body 7 has 1470 triangles, 0 quadrilaterals.
    Body 8 has 898 triangles, 0 quadrilaterals.
    Body 9 has 2248 triangles, 0 quadrilaterals.
    Body 10 has 1966 triangles, 0 quadrilaterals.
    Body 11 has 416 triangles, 0 quadrilaterals.
    Body 12 has 2268 triangles, 0 quadrilaterals.
    Body 13 has 17273 triangles, 10683 quadrilaterals.
    Body 14 has 17103 triangles, 11474 quadrilaterals.
#Enter Fieldview file mesh scaling (multiplication) factor >1
```

The existing solution parameters in the Fieldview file are listed as possible functions to the user. In addition, there are derived functions, such as Mach number or velocity magnitude, that are also listed as possible functions to the user. This list of functions is somewhat dynamic and changes depending on the original data supplied in the Fieldview file. An example of this list is shown below for a plot file created by the SimCenter solver Tenasi.

```
    Adaptation function 1 - Velocity magnitude
    Adaptation function 2 - Pressure
    Adaptation function 3 - Mach number
    Adaptation function 4 - Vorticity magnitude
    Adaptation function 5 - X vorticity magnitude
    Adaptation function 6 - Y vorticity magnitude
```

```
Adaptation function 7 - Z vorticity magnitude
Adaptation function 8 - Helicity
Adaptation function 9 - Density
Adaptation function 10 - X_Momentum; Momentum
Adaptation function 11 - Y_Momentum
Adaptation function 12 - Z_Momentum
Adaptation function 13 - TotalEnergy
Adaptation function 14 - Temperature
Adaptation function 15 - Distance_Function
Adaptation function 16 - Eddy_Viscosity
Adaptation function 17 - Part
#Enter adaptation function [0 to exit] >
```

For the function list shown, the first eight functions are derived from the data contained in the Fieldview file. Functions 9 through 17 are actual solution functions listed in the Fieldview file. Depending on the solution parameters contained in the Fieldview file the derived functions will change. For instance, if pressure was not one of the original functions it would be computed using density and temperature. Some solvers output pressure and density to the file, so temperature would be computed. Once a function is selected the users is asked whether the prism nodes are included in the computation. If the function were velocity magnitude or Mach number, including the prism nodes in the boundary layer may skew the sizing field significantly. Remember the viscous layers are intended to be inserted after the adaptive refinement, so excluding the prism nodes for those functions is recommended.

```
#Enter adaptation function [0 to exit] >2

Adaptation function is pressure.
#Include viscous (prism) nodes in the process? [0 1] >1

#Enter length scale exponent [>= 1.0] >2

#Enter nodal gradient weighting [0 - volume, 1 - inverse distance] >0

Statistics for adaption function 2 - Pressure:
# of edges= 2956978
  Min       = 0
  Mean      = 0.000562208
  Max       = 0.125329
  STD       = 0.0020037


#Enter threshold (+ve) or # of STD above mean (-ve) >-.5

  Threshold = 0.00156406
```

The length scale exponent is used with the edge length in the adaptation function shown in equation 2. The computed function at the edge is the dot product of the gradient of the selected function and the normalized edge vector, all multiplied by the length of the edge raised to the power p.

$$AF = \left(\frac{\left|\nabla f_0 \bullet \hat{e}\right| + \left|\nabla f_1 \bullet \hat{e}\right|}{2} \right) l^p \qquad (2)$$

where subscript 0 and 1 correspond to the endpoint nodes of the edge. The direction of the edge is $\hat{e}$ and the length of the edge is l. Raising the length of the edge to a power ensures that shocks will not dominate the refinement process. A typical value of this exponent p is 2. The user can also choose between volume weighting and inverse distance weighting for the Green-Gauss gradient calculation at the nodes using the next input variable.

Statistical data is then displayed and the user enters either an absolute threshold level (positive) or a multiplier on the standard deviation (negative). This threshold, AF_T, is

use with the computed adaptation function to back out a desired spacing value, equation 3.

$$d = \sqrt[p]{\frac{AF_T}{|\nabla f \bullet \hat{e}|}} \qquad (3)$$

Equation 3 is basically equation 2 solved for l, which is labeled as d. An edge with the adaptation function equal to the threshold will produce a spacing value equal to its length. Edges with the adaptation greater than the threshold will produce a spacing value smaller than the current length, etc.

Next the user is given the opportunity to select another function. Any new size computed for an edge will only be saved if the value is smaller than the current saved value. If no other functions are selected the user is asked whether to store all scalars, all tensors or a mix of both. The sizing parameters can be stored at the nodes or the elements of the Fieldview mesh. If stored at the nodes a bounding box surrounding the node that is roughly size to extend half way to the neighboring nodes is computed and store with the scalar or tensor. The user then has the option to manually enter additional size parameter boxes before the file is written and SPACING completes.

```
#Store scalars, tensors or mixed? [0 1 2] >2

#Store at nodes [0] or elements? [1] >0

Nodes have been processed, # of entries = 80397
   # of scalars stored = 19528
   # of tensors stored = 293085
Filename = <adapt.hdf5.fv>
#Enter an auxiliary scalar/tensor box? [0 1] >0

Tensor file written.
```

The tensors stored in the file represent the desired spacing field at each node or element contained in the input Fieldview file. This tensor is constructed using unit vectors representing the principle directions and the desired spacing values in those directions.

$$M = \left[R\right]\left[\lambda\right]\left[R\right]^{-1}$$

$$\text{2-D} \rightarrow M = \begin{bmatrix}\vec{e}_1 & \vec{e}_2\end{bmatrix}\begin{pmatrix} h_1^{-2} & 0 \\ 0 & h_2^{-2}\end{pmatrix}\begin{bmatrix}\vec{e}_1^{\,T} \\ \vec{e}_2^{\,T}\end{bmatrix}$$

$$\text{3-D} \rightarrow M = \begin{bmatrix}\vec{e}_1 & \vec{e}_2 & \vec{e}_3\end{bmatrix}\begin{pmatrix} h_1^{-2} & 0 & 0 \\ 0 & h_2^{-2} & 0 \\ 0 & 0 & h_3^{-2}\end{pmatrix}\begin{bmatrix}\vec{e}_1^{\,T} \\ \vec{e}_2^{\,T} \\ \vec{e}_3^{\,T}\end{bmatrix} \qquad (4)$$

Equations 4 shows how these tensors are computed in 2D and 3D, given the principle directions, $\vec{e}_1, \vec{e}_2, \text{and } \vec{e}_3,$ and the corresponding spacing values h_1, h_2, and h_3.This tensor can be used to compute the metric length of an edge in the mesh using equation (1), where AB is a vector defined from node A to node B. If the value of d_{AB} is unity the edge is at the prescribed length according to the tensor M. Spacing uses the direction of the function gradient as the first principle direction and constructs two additional directions that are orthonormal to the first.

4.5.1 OCTREE Library

Part of the source files delivered with the SPACING program is a standalone data storage library based on an Octree data structure. This library is used within the suite of meshing codes to store the discrete geometry and is used by the SPACING FIELD library, discussed below, to store the spacing field data. The geometry for the meshing codes is specified as a triangulated surface mesh. This mesh is stored by P_HUGG, P_OPT and P_VLI in an Octree data structure. Each facet is stored in a level of the tree. Queries of the geometry, such as closest point projection, are performed by searching the tree from the top level down. Only facets down branches of the tree in the vicinity of the query point are tested for operations such as closest point projection or intersection. The OCTREE library is made general by storing pointers to the actual entity that is stored. When spacing field data is stored in an Octree the pointers to the C++ class that contain the spacing data are stored in the tree. In either case, geometry data or spacing data, a bounding box for the stored entity is also stored with the pointer, which is used by the OCTREE library to perform the proximity tests.

This library is very flexible and can be managed in a dynamic fashion. Entries in the library can be added, removed or replaced by the calling program. An example of the use of the OCTREE library can be found in the source file for the solution interpolation program INTERP.

4.5.2 SPACING FIELD Library

Spacing field data is stored and retrieved via a SPACING FIELD library. This library is callable from C/C++. The library is used by P_HUGG, P_OPT and P_REFINE. It can also be called from any other meshing programs that require sizing information to perform mesh adaptation/refinement. The interface to the library has very few user calls, most of which deal with returning a size parameter given a location in space. Users can request the size parameter for a point or an edge. They can also request a Riemannian metric tensor for a point or an edge. Also included in the library are functions that return the metric length of an edge. In additions there are a few calls that allow the user to manipulate a Riemannian metric tensor. One call can create the tensor given three principal directions and desired spacing values in those directions. Another call can decompose the tensor into the principal directions and associated spacing values. There is an initialization call that sets up the library given a the name of the file that contains the spacing data, as created by the program SPACING. And there is a finalize call that closes down the library and releases the memory back to the system. An example of the use of these call can be found in the source for P_REFINE. All SPACING FIELD library routines begin with SF_ in the call name.

4.6 CONV

The convert utility code serves two functions; file conversion and parallel partitioning and recombination. The program options are displayed as a menu to the user. The first function, option 1, allows for the conversion to and from several common file formats into the CGNS file format used by the suite of meshing codes. When option 1 is selected CONV displays a list of supported file formats and whether the file format can be an input file, an output file or both. The user enters the input and output file names. The specific type of conversion is triggered by the extension of the filenames. A sample session, which converts the CGNS files used by the mesh codes to a Fieldview binary file, is shown below.

```
Conv:
#Exit              - 0
#Convert file      - 1
#Decomp CGNS file  - 2
#Recomp CGNS files - 3
```

```
#Choice >1

Supported Input File Formats          Input/Output  File Extension
------------------------------------ ------------  --------------
Fieldview ASCII double precision         both       .crunch
Fieldview binary                         both       .uns or .fv
Ensight                                 output      .case
CGNS binary                              both       .cgns
Generic mesh ASCII double precision      both       .mesh3D
CART3D boundary file                    output      .tri

#Enter input grid file name >test.cgns

#Enter output grid file name >test.fv

#Check volume element winding? [0,1] >0

#Check boundary element winding? [0,1] >0


Input mesh statistics:
 # of nodes = 60333
 # of tetrahedra = 335907
 # of boundaries = 7
Boundary 1 name = Xmin
  # of triangles      = 340
Boundary 2 name = Xmax
  # of triangles      = 340
Boundary 3 name = Ymin
  # of triangles      = 340
Boundary 4 name = Ymax
  # of triangles      = 340
Boundary 5 name = Zmin
  # of triangles      = 340
Boundary 6 name = Zmax
  # of triangles      = 340
Boundary 7 name = Sphere
  # of triangles      = 16624
Total number of surface triangles      = 18664


Filename = <test.fv>

Conv:
#Exit              - 0
#Convert file      - 1
#Decomp CGNS file  - 2
#Recomp CGNS files - 3
#Choice >0
```

When the output filename is a .tri file then an additional filed is created, which is the geometry params file used by the suite of meshing codes. The filename for this params files is hard-coded to “P_HUGG_geometry.params”. If the file already exists it will be overwritten. The volume and boundary element winding tests were created for tetrahedral meshes created by Pointwise to check for proper connectivity of tetrahedral and surface triangles. It is not necessary when converting to and from files created by the suite of meshing programs developed under this contract.

The second option of CONV is used to partition a CGNS file for parallel operations within the suite of meshing codes. The input file must be a CGNS file created by the meshing codes or converted from another source using CONV. The user inputs the number of partitions and then selects one of two partitioning strategies. The first partitioning strategy is MeTis and the second is a method based on a split-tree subdivision scheme. An output file is written for each partition where the prefix of the filename has an underscore and the partition number appended. The file suffixes are unchanged. A sample session for partitioning the file test.cgns into 4 partitions is shown below.

```
Conv:
#Exit              - 0
#Convert file      - 1
#Decomp CGNS file  - 2
#Recomp CGNS files - 3
#Choice >2

#Enter input serial CGNS file name >test.cgns

#Enter number of partitions >4

#Partitioning model [0 - metis, 1 - split tree] >1

Using balanced split-tree partitioning for 60333 nodes.
Current number of partitions = 1
Splitting partition 0 at Z = -0.0113646
Current number of partitions = 2
Splitting partition 1 at Z = 0.676204
Current number of partitions = 3
Splitting partition 0 at Z = -0.695569
Partition 0 has 15083 vertices
Partition 1 has 15083 vertices
Partition 2 has 15084 vertices
Partition 3 has 15083 vertices
Partitioning complete.

Writing parallel file <test_0.cgns>
 Number of nodes = 17443
 Number of tetrahdedra = 92200
 Boundary 1, number of triangles      = 171
 Boundary 2, number of triangles      = 169
 Boundary 3, number of triangles      = 169
 Boundary 4, number of triangles      = 171
 Boundary 5, number of triangles      = 340
 Boundary 7, number of triangles      = 2664
Writing parallel file <test_1.cgns>
 Number of nodes = 19383
 Number of tetrahdedra = 94071
 Boundary 1, number of triangles      = 34
 Boundary 2, number of triangles      = 21
 Boundary 3, number of triangles      = 21
 Boundary 4, number of triangles      = 34
 Boundary 7, number of triangles      = 6018
Writing parallel file <test_2.cgns>
 Number of nodes = 17419
 Number of tetrahdedra = 91858
 Boundary 1, number of triangles      = 171
 Boundary 2, number of triangles      = 175
 Boundary 3, number of triangles      = 175
 Boundary 4, number of triangles      = 171
 Boundary 6, number of triangles      = 340
 Boundary 7, number of triangles      = 2821
Writing parallel file <test_3.cgns>
 Number of nodes = 19421
 Number of tetrahdedra = 94427
 Boundary 1, number of triangles      = 25
 Boundary 2, number of triangles      = 35
 Boundary 3, number of triangles      = 35
 Boundary 4, number of triangles      = 25
 Boundary 7, number of triangles      = 5889

Conv:
#Exit              - 0
#Convert file      - 1
#Decomp CGNS file  - 2
#Recomp CGNS files - 3
#Choice >0
```

The serial CGNS filename is supplied to the suite of meshing codes. If the code is running in parallel then the underscore and partition number will be added by the code

before reading or writing the files. This allows the input decks to the codes to be unchanged whether running in serial or parallel.

The final option of CONV is used to recombine parallel CGNS files into a single file. The user inputs the serial file name. CONV will search the working directory for the parallel files in sequential order, based on the underscore and partition number in the filename prefix. CONV should automatically determine the correct number of partitions in the directory. If a case is partitioned, recombined and then partitioned again in the same directory then the previous partitioned files should be deleted before the second partitioning pass. This will prevent CONV from counting the incorrect number of partitions if the second partitioning pass used fewer processors than the first pass. A sample session of recombining partition files is shown below.

```
Conv:
#Exit              - 0
#Convert file      - 1
#Decomp CGNS file  - 2
#Recomp CGNS files - 3
#Choice >3

#Enter output serial CGNS file name >test.cgns

CGNS file <test_0.cgns> found.
CGNS file <test_1.cgns> found.
CGNS file <test_2.cgns> found.
CGNS file <test_3.cgns> found.
CGNS file <test_4.cgns> not found.
Number of CGNS part files detected = 4
File <test_0.cgns> information:
  Number of nodes = 17443
  Number of boundaries = 7
Boundary 1, number of triangles      = 171
Boundary 2, number of triangles      = 169
Boundary 3, number of triangles      = 169
Boundary 4, number of triangles      = 171
Boundary 5, number of triangles      = 340
Boundary 7, number of triangles      = 2664
  Number of tetrahedra = 92200
File <test_1.cgns> information:
  Number of nodes = 19383
  Number of boundaries = 7
Boundary 1, number of triangles      = 34
Boundary 2, number of triangles      = 21
Boundary 3, number of triangles      = 21
Boundary 4, number of triangles      = 34
Boundary 7, number of triangles      = 6018
  Number of tetrahedra = 94071
File <test_2.cgns> information:
  Number of nodes = 17419
  Number of boundaries = 7
Boundary 1, number of triangles      = 171
Boundary 2, number of triangles      = 175
Boundary 3, number of triangles      = 175
Boundary 4, number of triangles      = 171
Boundary 6, number of triangles      = 340
Boundary 7, number of triangles      = 2821
  Number of tetrahedra = 91858
File <test_3.cgns> information:
  Number of nodes = 19421
  Number of boundaries = 7
Boundary 1, number of triangles      = 25
Boundary 2, number of triangles      = 35
Boundary 3, number of triangles      = 35
Boundary 4, number of triangles      = 25
Boundary 7, number of triangles      = 5889
  Number of tetrahedra = 94427


Serial file <test.cgns> information:
Total number of nodes = 60333
```

```
Total number of boundaries = 7
Boundary 1, total number of triangles      = 340
Boundary 2, total number of triangles      = 340
Boundary 3, total number of triangles      = 340
Boundary 4, total number of triangles      = 340
Boundary 5, total number of triangles      = 340
Boundary 6, total number of triangles      = 340
Boundary 7, total number of triangles      = 16624
Total number of tetrahedra = 335907
Writing test.cgns

Conv:
#Exit              - 0
#Convert file      - 1
#Decomp CGNS file  - 2
#Recomp CGNS files - 3
#Choice >0
```

4.7 INTERP (Task 3.3)

Once a mesh has been refined the solution from the initial mesh can be transferred to the new mesh using INTERP. This program uses the OCTREE library to store the original mesh and solution and computes new solution variables at the nodes of the new mesh using a simple inverse-distance weighted interpolation scheme. The code only operates in serial mode. A parallel version was not developed as part of this contract.

The input and output mesh and solution files are in a Fieldview binary format. The input mesh for the case shown in Figure 11 was the Tenasi solution for the initial mesh created using Pointwise and P_VLI. The red contours are density contours for that mesh. The green contours are plotted on the mesh created by using P_REFINE with a spacing field on the initial mesh. The adaptation function was pressure. Qualitatively the solutions are almost identical.

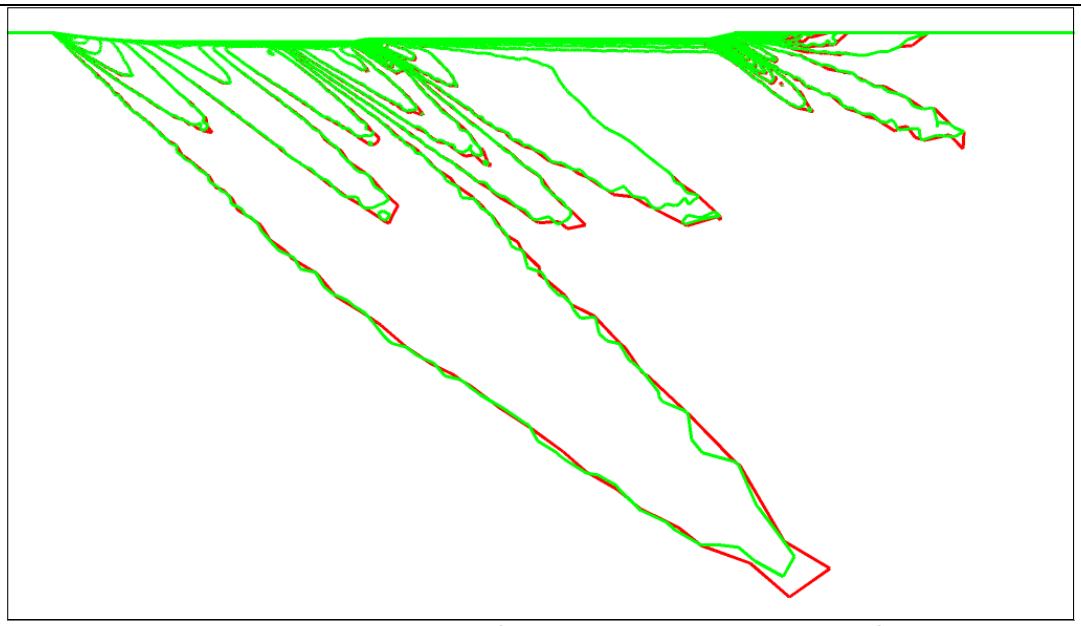


Figure 11. Density contour comparison for original mesh (red) and refined/interpolated mesh (green).

5.0 Sample Case Studies Using Supersonic Delta Wing dw4

The supersonic wind tunnel geometry described in Section 4 was used in two case studies which performed feature-based adaptive refinement. The geometry of the configuration is model 4, dw4, from the report in reference 8. Since the configuration is symmetric from side to side and from top to bottom and an angle of attack zero degrees was chosen only one fourth of the configuration was modeled. The extents of the domain are shown in Figure 12 with the dw4 situated along the upper left edge of the box. Figure 13 shows a close up view of the dw4, shown in gray, with the sting shown in green. The geometry of the sting was fabricated in Pointwise and does not represent the actual sting geometry from the wind tunnel test. The reason for including the sting is to provide some level of detail at the base of the dw4 geometry that does not adversely affect the pressure signature at the bottom boundary. The extents of the domain are such that the bottom boundary is at 3.6 body lengths from the vehicle. This is the location where the pressure measurements were performed, so pressure levels on the bottom boundary can be compared with the test results.

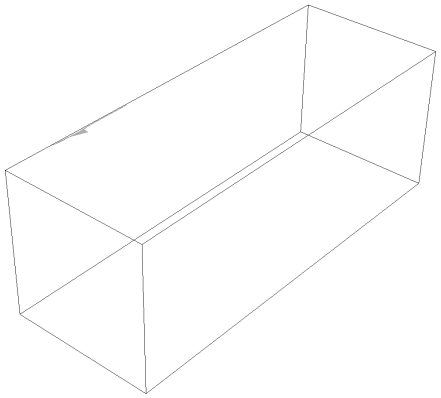


Figure 12. Extents of computational domain with vehicle shown along top left edge.

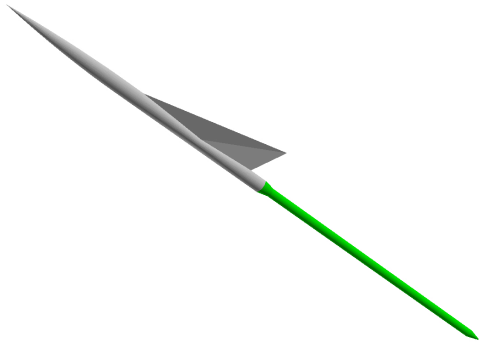


Figure 13. Close up view of dw4 with sting.

This geometry was analyzed using the SimCenter Navier-Stokes flow solver Tenasi on two different mesh types. The first mesh was a hexahedral-dominant inviscid mesh created in P_HUGG. Viscous layers were then inserted using P_VLI. The second mesh was a tetrahedral mesh created in Pointwise. Similarly, viscous layers were inserted using P_VLI. Solution-based adaptive refinement was performed on both meshes using the SPACING program to define the spacing field. P_HUGG used that spacing field in subsequent meshes to adapt to the shocks in the solution in creating a new inviscid mesh. P_REFINE used the spacing field to subdivide the inviscid tetrahedral mesh and adapt to the shocks from that Tenasi solution. For both cases viscous layers were reinserted using P_VLI. Details of each process will be described below.

The resulting viscous meshes were analyzed using the SimCenter Navier-Stokes code Tenasi for Mach 1.68 flow at a Reynold's number of 4.13e06. The experimental report did not contain enough flow conditions to accurately determine the proper Reynold's number. The value chosen corresponded roughly to sea level conditions. This configuration and flow conditions does not require viscous analysis to compute accurate pressure signatures. The intent of these analyzes was to demonstrate the use of

adaptive refinement on viscous flowfields, so an approximate Reynold's number was sufficient. A one-equation k-ε turbulence model was used for all solutions.

The resulting pressure signatures at the bottom boundary for each grid sequence are compared to the wind tunnel results.

5.1 P_HUGG Mesh with Viscous Layers

The geometry parameter file for the initial P_HUGG mesh is shown below. The layers flag is set to 1 for all the vehicle body parts. This instructs P_HUGG to perform a curvature-based adjustment to the initial spacing parameter, g_space. The n_space value is not used since the Layers flag was not set to 2 or 3.

```
Number of boundary files
1
dw4_two_sym.tri
Number of boundaries
14
Boundary #   Layers      g_space       n_space              boundary name
---------- ---------- -------------- -------------- --------------------------------
         0          0         -5e+18         -5e+18 Virtual
         1          1     0.05000000          5e-05 Bevel
         2          1     0.10000000          5e-05 Body_LL
         3          1     0.05000000          5e-05 Cone
         4          1    0.001296997          5e-05 Cone_Base
         5          0       5.000000       1.328125 Far_bottom
         6          0      10.000000       1.328125 Far_inflow
         7          0      10.000000       1.328125 Far_left
         8          0      10.000000       1.328125 Far_outflow
         9          1     0.10000000          5e-05 LL1
        10          1     0.10000000          5e-05 LL2
        11          1   8.106232e-05          5e-05 Nose
        12          1     0.20000000          5e-05 Sting
        13          0       1.328125       1.328125 Y_sym
        14          0       1.328125       1.328125 Z_sym
```

The input deck to P_HUGG is shown below. No adaptation file is specified for this initial mesh. P_HUGG was run on a Linux workstation using two processors. The wall clock time was 200.58 seconds. The initial inviscid contained 365,963 nodes, 522,007 tetrahedra, 424,891 pyramids and 100,027 hexahedra.

```
#Number of geometry files >
1
#File 1 >
dw4_two_sym.tri
#Adaption spacing file present? [0=no, 1=yes] >
0
#Enter minimum spacing ->
0.05
#Enter maximum spacing ->
10
#Enter maximum X-Y aspect ratio ->
1
#Enter maximum X-Z aspect ratio ->
1
#Enter maximum Y-Z aspect ratio ->
1
#Enter cutting tolerance [ -ve multiplier, +ve absolute] ->
1e-10
#Enter boundary face merging tolerance [ degrees ] ->
20
#Enter boundary collinear point merging tolerance [ degrees ] ->
20
#Allow polygonal faces? [0=no, 1=yes] ->
0
#Allow polyhedra? [0=no, 1=yes] ->
```

```
0
#Allow surface quads? [0=no, 1=yes] ->
0
#Enter spawn level ->
3
```

P_OPT was used to smooth the surface nodes and 3 layers of interior nodes. The cost threshold was 0.1 and 100 iterations were used to reduce the maximum cost from 1.0 down to 0.7299. Initially there were 19 inverted elements that were corrected by the 4^{th} iteration.

The geometry parameter file was not changed when P_VLI was used to insert viscous layers. The initial spacing was set to 5e-05 for the vehicle body parts. The input deck to P_VLI is shown below. The layer-by-layer approach was used to insert a maximum of 25 viscous layers. The code was run on 8 processors on a Linux cluster. The total wall clock time was 4 hours and 2 minutes. The final viscous mesh contained 1,662,447 nodes, 587,546 tetrahedra, 503,402 pyramids, 2,397,967 prisms and 121,572 hexahedra.

```
#Number of geometry files ->
1
#File 1 ->
dw4_two_sym.tri
#Enter physical grid file name, serial version ->
vr1.cgns
#Enter number of viscous layers to insert >
25
#Enter 1st pass # of outer smoothing sweeps >
1
#Enter 2nd pass # of outer smoothing sweeps >
10
#Enter # of Linear-Elastic smoothing sweeps (updated each time) >
50
#Enter stiffness model [0-CN, 1-CN/vol, 2-Unity] >
0
#Enter # of normal smoothing sweeps >
10
#Enter convergence OoM [ > 0] >
3
#Enter # of GMRES search directions [ > 0] >
30
#Enter initial geometric progression factor [1.0-1.5] >
1.15
#Enter geometric progression growth factor [1.0-1.5] >
1.01
#Enter minimum geometric progression factor [1.0-1.5] >
1.15
#Enter maximum geometric progression factor [1.0-1.5] >
1.25
#Overwrite restart at each layer [0 1] >
1
```

A plot of the viscous mesh is shown in Figure 14. The matching of the spacing of the outer viscous layer with the interior mesh is better than the matching for the all-at-once approach shown in Figure 8. So this mesh was slightly larger than the mesh in Figure 8 and took about 3 hours longer due to the layer-by-layer approach.

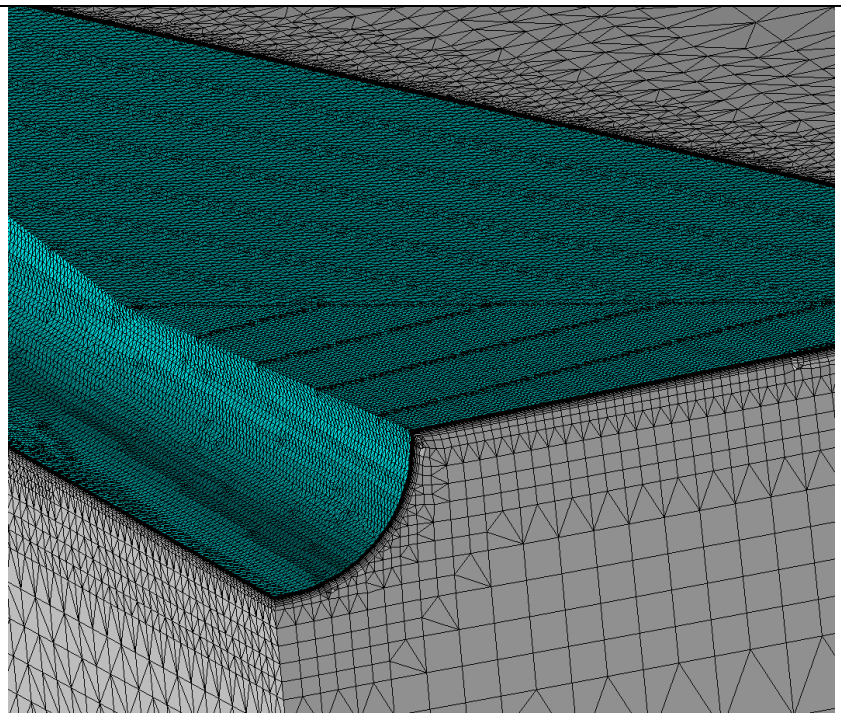


Figure 14. Viscous layers inserted in the P_HUGG mesh using the layer-by-layer approach.

A viscous solution was computed using Tenasi. The contours of delta pressure are shown in Figure 15. The range of the contours was set to correspond with the minimum and maximum values at the bottom outer boundary where the measurements were made. It is obvious from the figure that the shock structure is grossly dissipated in the solution on this coarse mesh, since the bottom boundary is entirely green.

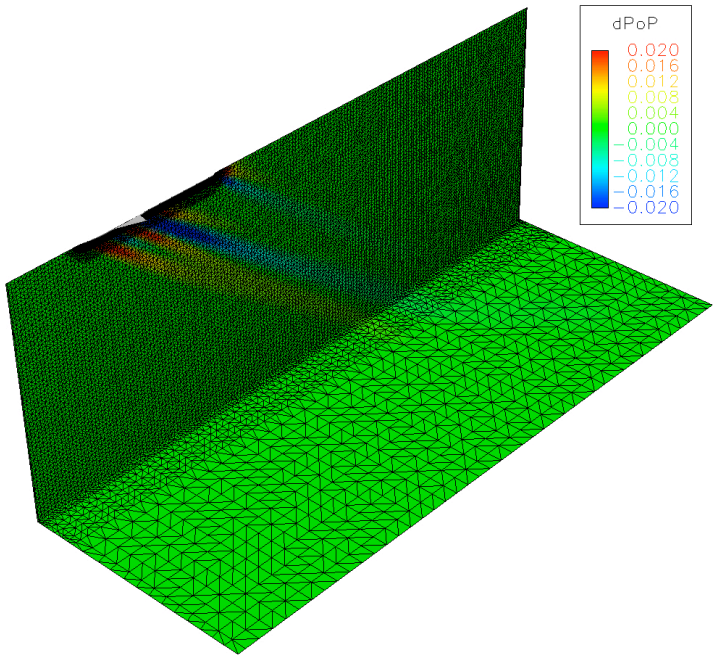


Figure 15. Delta pressure contours for initial solution using P_HUGG mesh.

The solution file was used with SPACING to create a spacing field for the next mesh. The selected functions were pressure and Mach number. The prism nodes were included in the spacing field calculations for pressure, but were excluded for Mach number. P_HUGG used that spacing field with the same input parameters to create the second mesh. P_OPT was used to smooth that inviscid mesh and P_VLI was used to reinsert viscous layers using the same approach. This process was repeated three times, resulting if four meshes total. The final mesh is shown in Figure 16 and Figure 17. It contained 28,984,962 nodes, 13,538,584 tetrahedra, 8,950,911 pyramids, 495,537 prisms and 23,458,606 hexahedra.

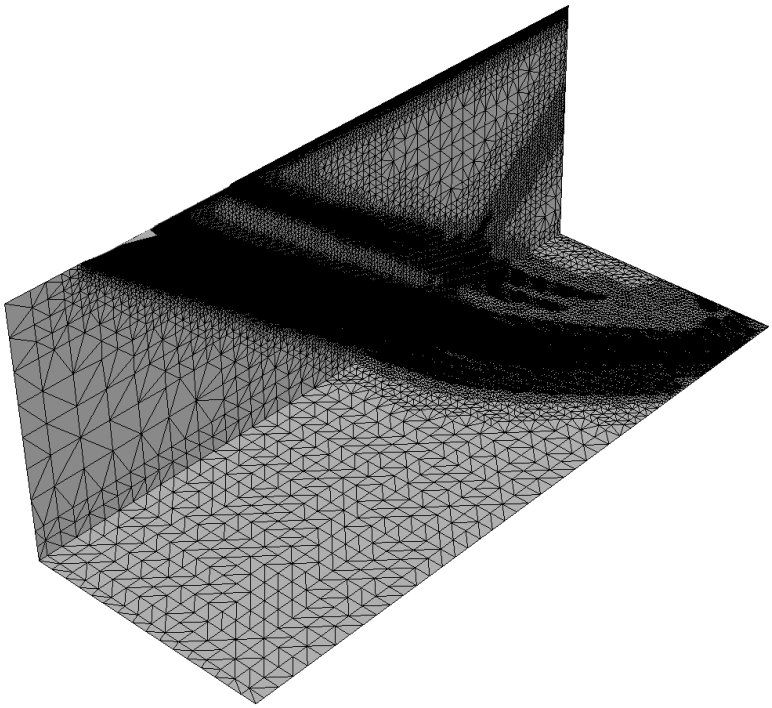


Figure 16. Final adapted P_HUGG mesh on Y symmetry plane and bottom outer boundary.

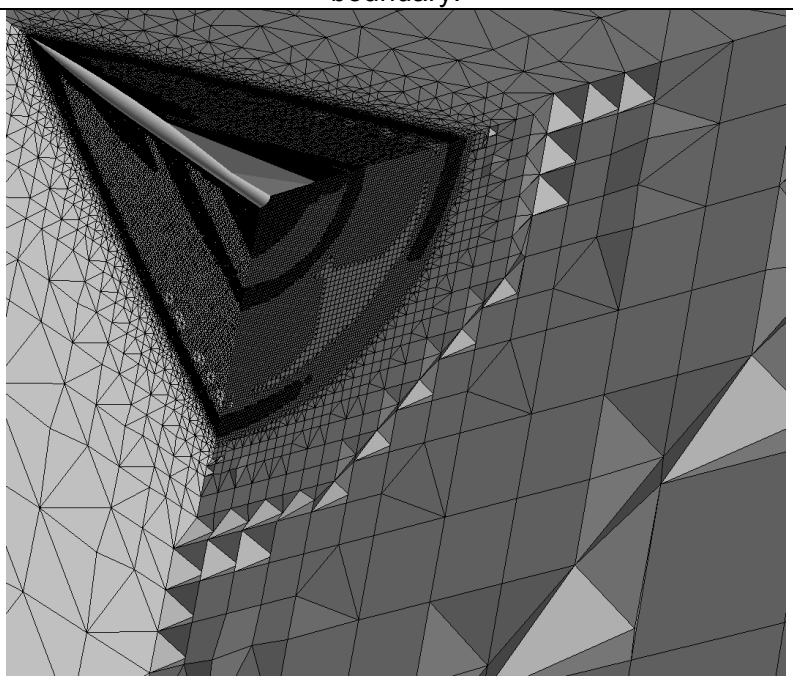


Figure 17. Adapted mesh on Y and Z symmetry planes and an X=15 cutting plane.

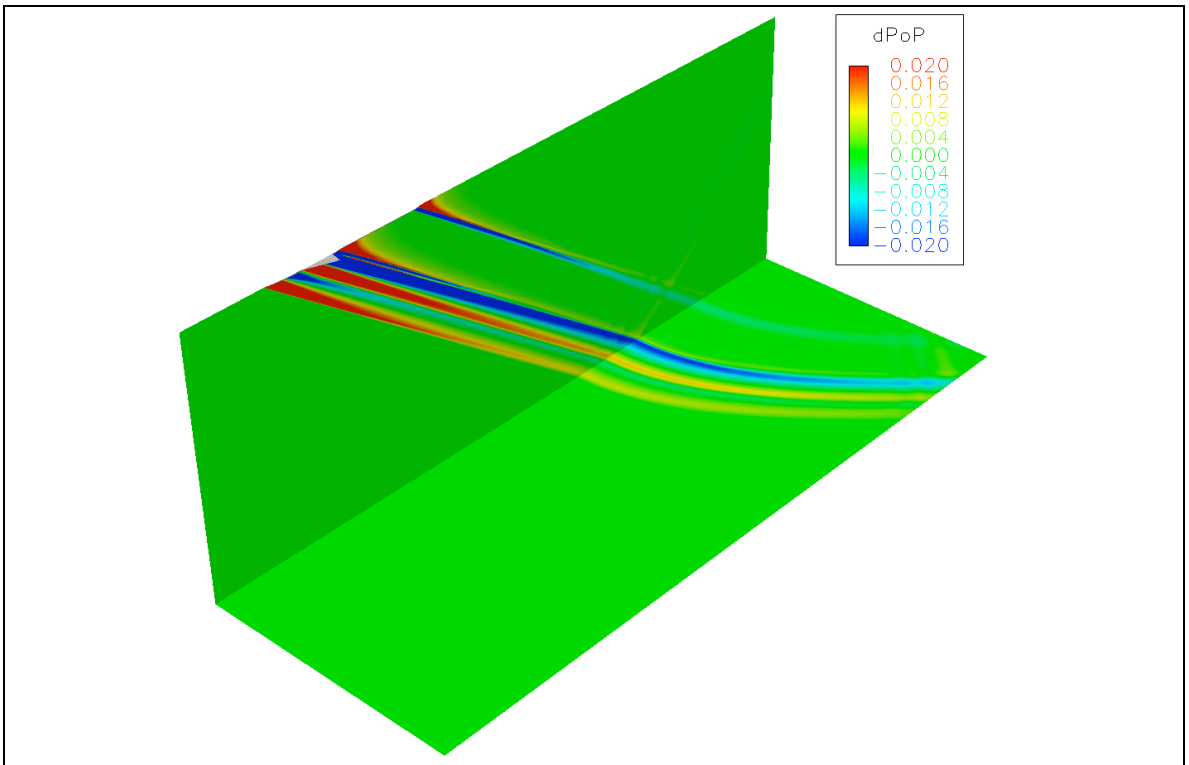


Figure 18. Final delta pressure contours on Y symmetry plane and bottom outer boundary.

The final solution is shown in Figure 18. The same contour range is used as shown in Figure 15. The pressure variation on the bottom outer boundary is now apparent in the contour plot. The pressure signature at the bottom boundary along the Y symmetry plane for each grid is compared to the experimental values in Figure 19. Grid 1 showed only a small variation in delta pressure, while Grid 4 matches the data very well. There is a slight under-prediction for the first two peaks and a small over-prediction for the third peak for the Grid 4 solution.

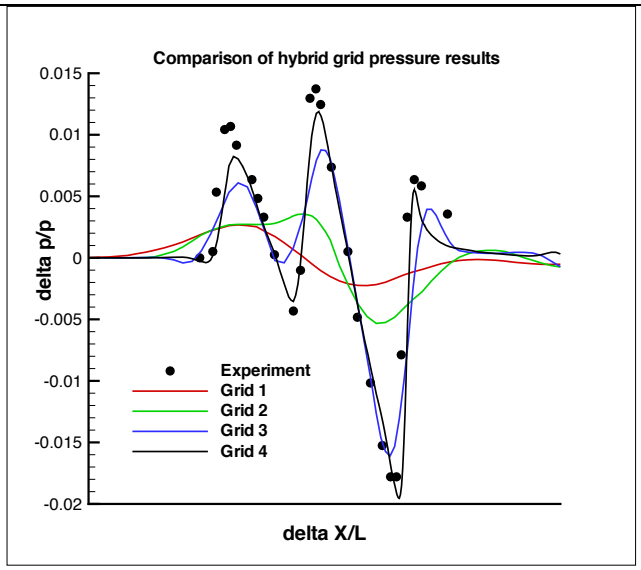


Figure 19. Comparison of delta pressures for each P_HUGG grid compared to wind tunnel measurements.

Table 1 shows the total number of nodes and elements for the P_HUGG meshes. The viscous layers for the last mesh were created using the all-at-once approach which resulted in considerably fewer prisms. Otherwise the element counts increased in a predictable way.

Table 1. Node and Element Counts for P_HUGG meshes

	# Nodes	# Tetrahedra	# Pyramids	# Prisms	# Hexahedra
Grid 1	1662447	587546	503402	2397967	121572
Grid 2	3272986	1808583	1411433	2397978	1203115
Grid 3	15057192	7402561	5179078	7878202	7941206
Grid 4	28984962	13538584	8950911	495537	23458606

5.2 Tetrahedra Mesh with Prismatic Viscous Layers Reinserted

The inviscid mesh for the second case was an all-tetrahedral mesh created using Pointwise. That mesh contained 21,811 nodes and 94,384 tetrahedra. The number of triangles on the boundaries was 23,216. The number of triangles on the vehicle was 10,006. P_VLI was used in the same manner described in Section 5.1 to insert viscous layers into this mesh. The code was run on 8 processors on a Linux cluster. The total wall clock time was 1 hours and 12 minutes. The final viscous mesh contained 312,613 nodes, 526,770 tetrahedra, 550 pyramids and 399,413 prisms. The resulting mesh is shown in Figure 20. This mesh is considerably smaller than the P_HUGG initial mesh.

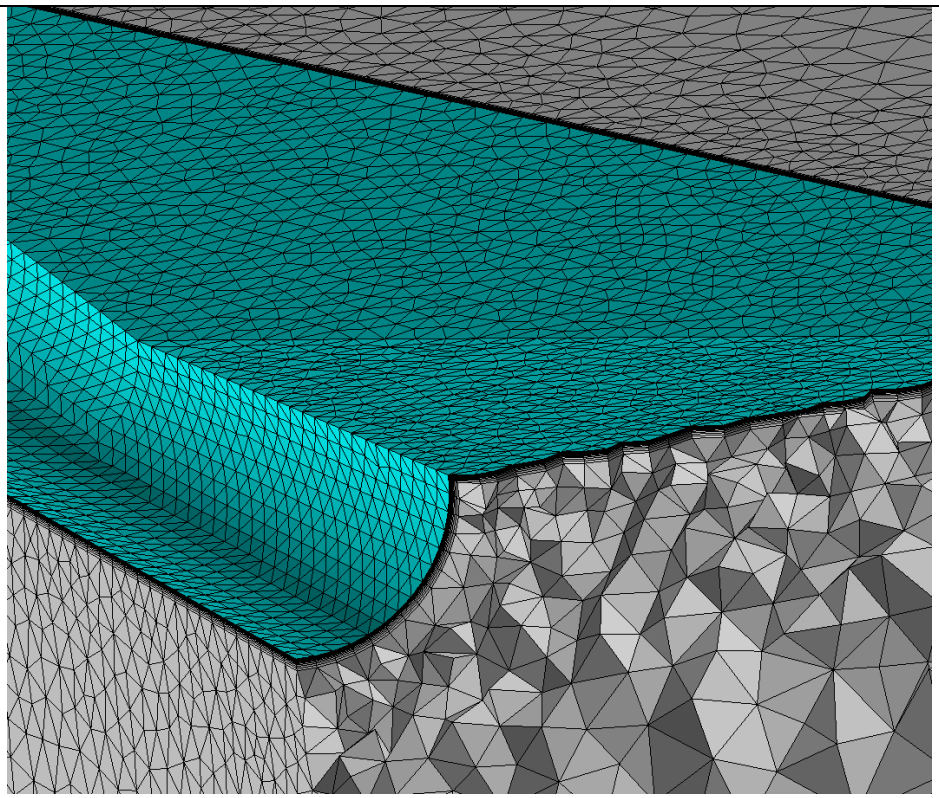


Figure 20. Viscous layers inserted into initial Pointwise mesh using P_VLI with layer-by-layer approach.

The initial solution computed using Tenasi is shown in Figure 21. The same contour range is shown in the plot. The extent of the shock structure is similar to the P_HUGG mesh result.

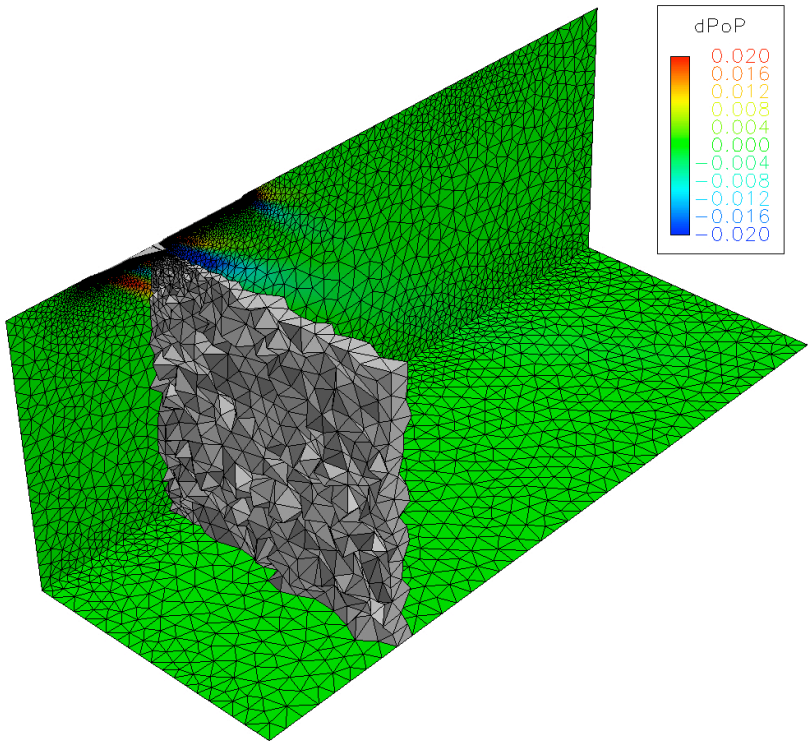


Figure 21. Initial Pointwise style mesh with delta pressure contours..

This solution file was used with SPACING to create a spacing field for the next mesh. The same adaptation functions were selected, pressure and Mach number. Similarly, the prism nodes were used with the pressure field and were not used with the Mach number field.

P_REFINE was used to refine the initial inviscid Pointwise mesh using the spacing field from SPACING. The target number of new nodes was purposely set large so the refinement was not restrained. Triangles were forced to subdivide into sub-triangles and tetrahedral were forced to subdivide into sub-tetrahedra. The average metric length of the initial mesh was 1.13991. The maximum metric length was 14.4631. Viscous layers were re-inserted using P_VLI with the same input parameters. A new solution was computed using Tenasi and the process was repeated 4 times, resulting in 5 meshes total. The final viscous mesh contained 32,708,453 nodes, 181,538,890 tetrahedra, 104,266 pyramids and 3,315,965 prisms. The final delta pressure solution is shown in Figure 22. The same contour range is used. The pressure variation on the bottom boundary is now apparent. The final mesh is shown in Figure 23.

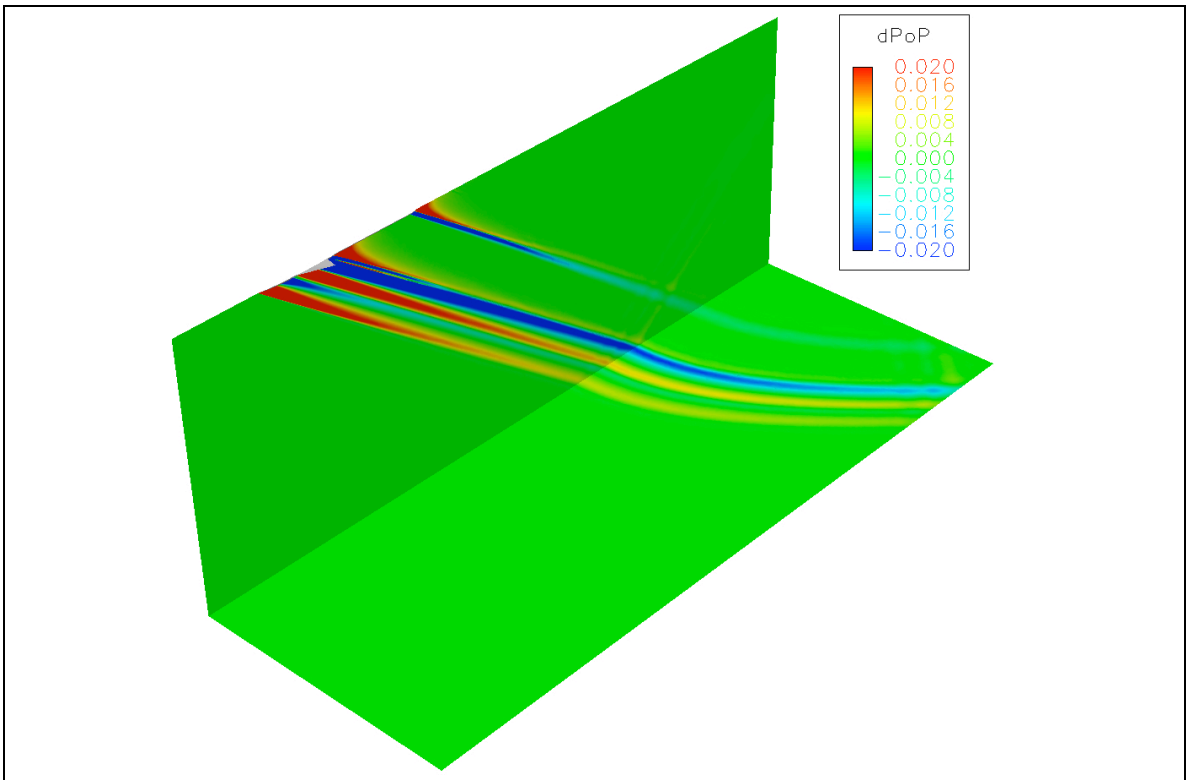


Figure 22. Final delta pressure contours on Pointwise style mesh on Y symmetry plane and bottom outer boundary.

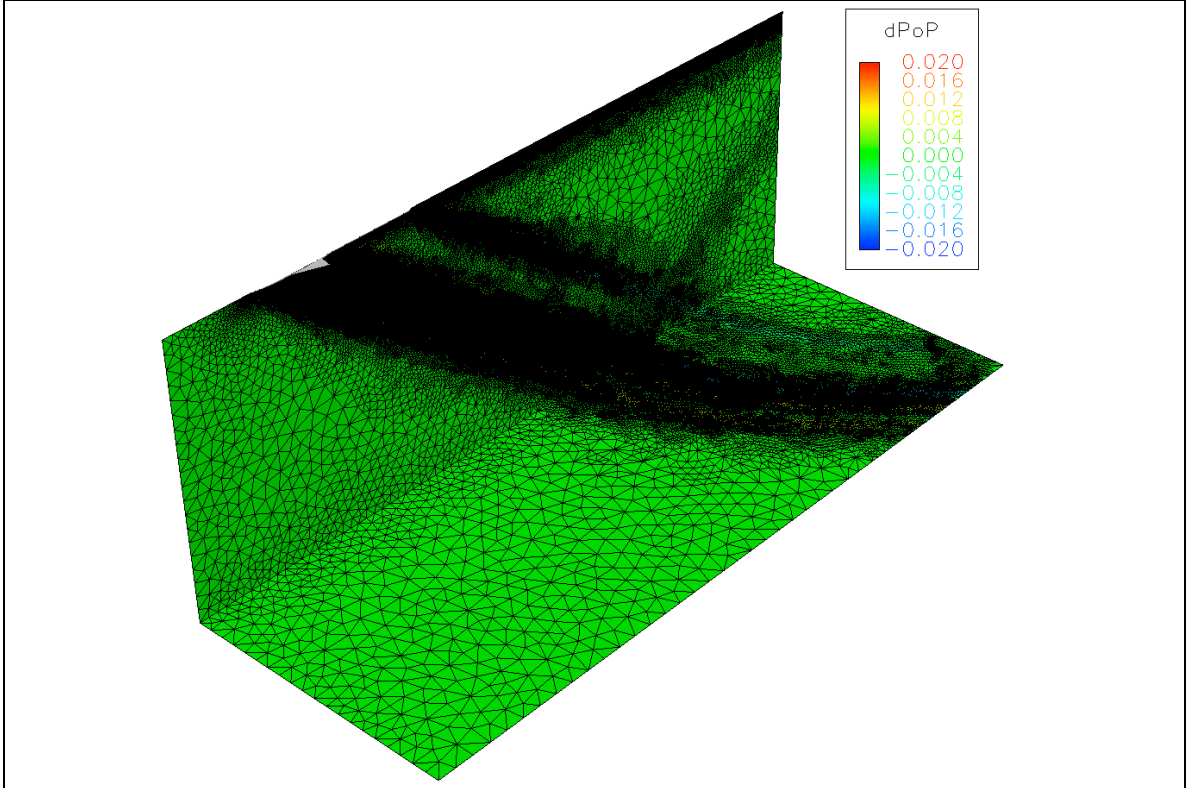


Figure 23. Final adapted Pointwise style mesh on Y symmetry plane and bottom outer boundary.

The solution and mesh is also shown in Figure 24 for the Y and Z symmetry plane and an X=15 cutting plane.

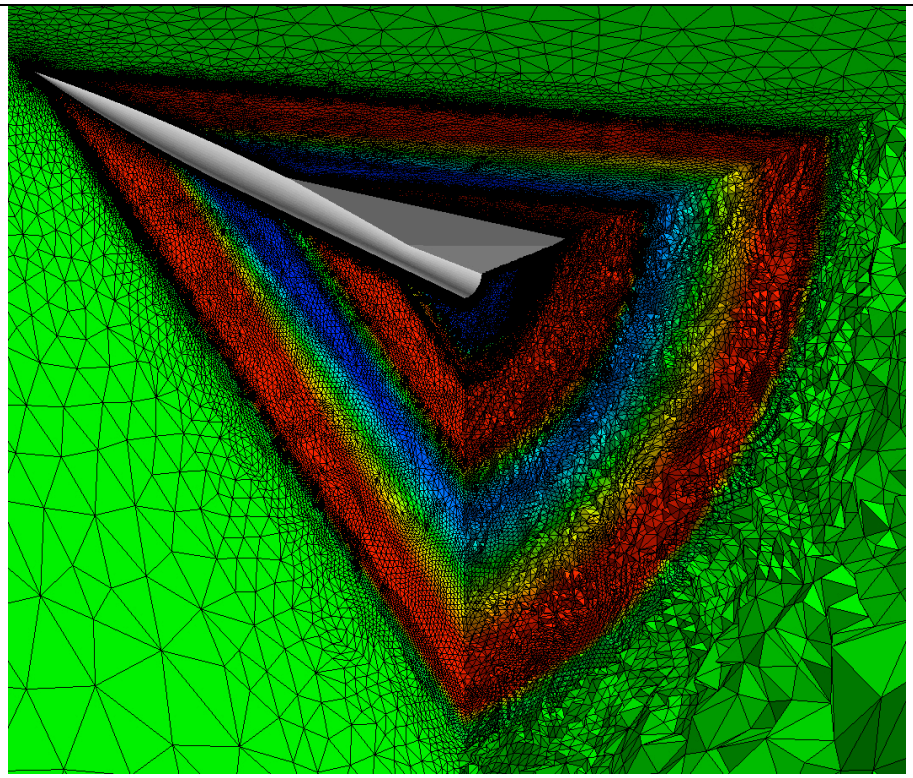


Figure 24. Final adapted mesh on Y and Z symmetry plane and at an X=15 cutting plane.

The pressure signature at the bottom boundary along the Y symmetry plane for each grid is compared to the experimental values in Figure 25. Grid 1 showed only a small variation in delta pressure, while Grid 5 matches the data much better. The under-prediction for the first two peaks is worse for this mesh than was shown for the P_HUGG mesh. The third peak for the Grid 5 solution matches the experimental data well. Four refinement passes were used to achieve this result. Perhaps additional passes can produce the accuracy achieved with the P_HUGG meshes.

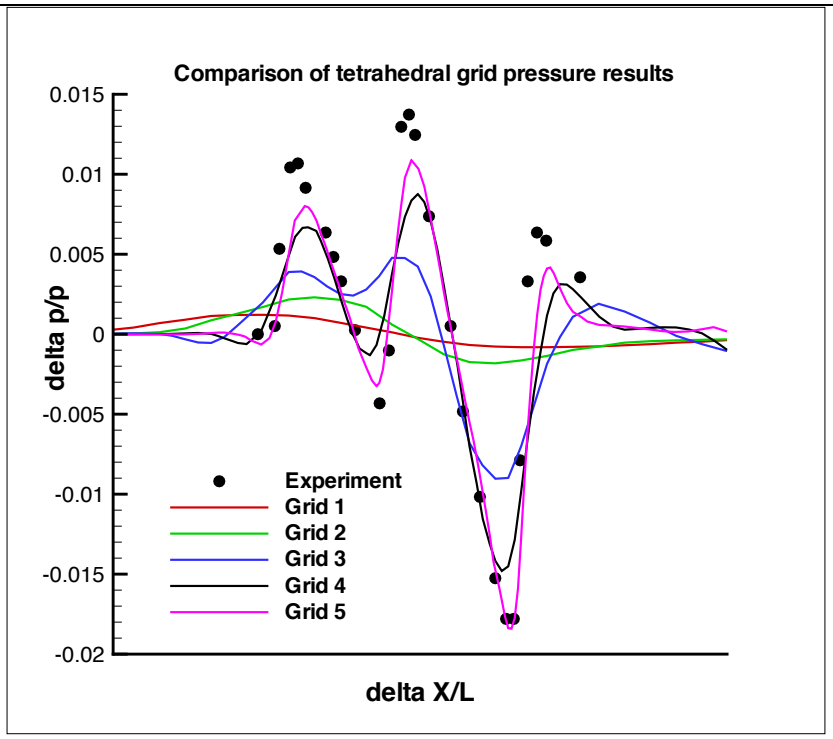


Figure 25. Delta pressure comparison for each Pointwise style mesh with experimental measurements.

Table 2 shows the total number of nodes and elements for the Pointwise style meshes. The viscous layers for the last mesh were also created using the all-at-once approach which resulted in fewer prisms. Otherwise the element counts increased in a predictable way. No hexahedra were created during the meshing and adapting process. The final Pointwise style mesh had roughly 4 million more nodes than the P_HUGG style mesh.

Table 2. Node and Element Counts for Pointwise style meshes

	# Nodes	# Tetrahedra	# Pyramids	# Prisms	# Hexahedra
Grid 1	312613	526770	550	399413	0
Grid 2	1050275	2467990	648	1174942	0
Grid 3	3708058	13139264	2993	2786601	0
Grid 4	12910643	63822730	10001	3922476	0
Grid 5	32708453	181538890	104266	3315965	0

Conclusions

The outcome of the research was an integrated system of tools that can be used for the unstructured viscous mesh creation and adaptation required for rapid high fidelity analysis of supersonic cruise vehicles. These grid tools include an inviscid mesh creation program, a mesh smoothing program, a program for creating an adaptation spacing field, a program to insert viscous layers into inviscid meshes and a mesh adaptation program that can refine a mesh to the prescribed spacing field. In addition, there are auxiliary programs that support the system by performing tasks such as mesh file format conversion, mesh partitioning and recombination and solution interpolation. Parallel processing is possible with all the major meshing tools.

The inviscid mesh creation program is P_HUGG, which is based on a Cartesian hierarchical approach. This program produces a hybrid mesh for arbitrary geometries comprised of tetrahedral, pyramids, prisms and hexes. It can be executed in serial or parallel. P_HUGG can accept a spacing field from the program SPACING to refine the inviscid mesh to match the desired spacing field. Elements at the geometry boundary are cut using a general cutting algorithm.

The mesh smoothing program is P_OPT. This code will operate in serial or parallel. It performs node perturbations in an attempt to optimize the position of the nodes to improve a mesh quality-based cost function. P_OPT is necessary for P_HUGG since the cutting process can create small sliver elements and inverted elements. This program is recommended for use on inviscid type meshes only, as the perturbation scheme and cost function will not respect the fine spacing associated with viscous meshes.

A spacing field can be created using the program SPACING. This program reads a Fieldview plot file generated by a flow solver. The user can select one or more adaptation functions that are either functions stored in the plot file or derived from the functions in the plot file. With these adaptation function SPACING will determine sizing information based on gradients of the adaptation function. This sizing field is stored at the nodes or elements of the Fieldview file mesh and output to a file for use by the other meshing tools.

Viscous layers can be inserted into an existing mesh using P_VLI. This program operates in serial or parallel. Linear-elastic mesh smoothing is used by P_VLI to move the existing mesh away from viscous boundaries, making room for layers of viscous prisms to be inserted. P_VLI can work with P_HUGG style meshes or traditional style unstructured meshes, such as those produced by Pointwise.

Mesh refinement is accomplished using P_REFINE. This tool is intended to be used with the Pointwise style meshes, not the P_HUGG style meshes. P_HUGG can perform adaptation itself, so P_REFINE is not necessary for that style of mesh. For traditional unstructured meshes containing tetrahedral, pyramids, prisms and hexes P_REFINE uses a spacing field created by SPACING to mark edges in the mesh for refinement. Edges and elements are then subdivided producing a new hybrid mesh.

One auxiliary program is CONV, which provides the ability to convert mesh file formats and partition and recombine meshes for the tools in the parallel processing environment. The partitioning uses MeTis or a split-tree structure to create partitioned files for the meshing tools.

Solutions data from a Fieldview file can be interpolated onto a refined mesh using the program INTERP. This tool uses the OCTREE library to store the data from the input mesh for rapid retrieval as it interpolates the solution onto the new mesh using an inverse distance weighted interpolation scheme.

Two additional software items were created during the contract and used by the other tools; an OCTREE storage system and a SPACING FIELD library. The OCTREE system can store data in a 3D spatial setting for rapid retrieval later. P_HUGG uses the OCTREE storage system to store the input geometry and to store the adaptive spacing field created by SPACING. P_OPT also uses the OCTREE system for similar storage needs. The SPACING field library is a C/C++ callable library that can be used by meshing programs to retrieve sizing information from a spacing field generated by SPACING. The spacing library is used by P_HUGG, P_OPT and P_REFINE. It also uses the OCTREE storage system internally.

The meshing tools created under this contract have been used daily in research activities at the SimCenter. Two examples are included in this report illustrating how the tools can be used to create and adapt viscous meshes for sonic boom predictions. One example used P_HUGG to create the inviscid mesh, P_OPT to smooth the mesh and P_VLI to add viscous layers. Viscous solutions obtained on these meshes were input into SPACING to create an adaptation spacing field. This spacing field was used in

subsequent runs of P_HUGG to create adapted inviscid meshes. Viscous layers were reinserted in the adapted meshes. A sequence of 4 grids was shown that increased the resolution of the shocks to improve the accuracy of the pressure signature at a measurement plane 3.6 body lengths from the vehicle. Comparisons with experimental data revealed increasing accuracy in the solution for each mesh adaptation pass.

The second example used a traditional style unstructured mesh to compute the same pressure signatures at the measurement plane. The initial inviscid mesh was an all-tetrahedral mesh created using Pointwise. P_VLI was used to insert viscous layers. Viscous solutions were obtained and used by SPACING to create an adaptation field. The inviscid tetrahedral mesh was then refined using this spacing field in the program P_REFINE. Output from P_REFINE was a new inviscid tetrahedral mesh. P_VLI was then used to reinsert viscous layers and the process was repeated. A sequence of 5 grids was created. As with the P_HUGG mesh, the accuracy of the predicted pressure signature increase with each mesh adaptation pass.

A data package is provided with this report that includes updated executables for P_HUGG, P_OPT and P_VLI. All other meshing tools are provided in source format with associate makefile to create the executables on a Linux system. Also included in the package are sample cases for the two types of meshes shown in Section 5.

References

1. Karman, S. L. Jr. and Betro, V., "Parallel Hierarchical Unstructured Mesh Generation with General Cutting", AIAA -2008-0918, Reno, NV, January 2008.
2. Karman, S. L. Jr., and Sahasrabudhe, M., "Unstructured Adaptive Elliptic Smoothing", AIAA-2007-0559, Reno, NV, January 2007.
3. Sahasrabudhe, Mandar, "Unstructured Mesh Generation and Manipulation Based on Elliptic Smoothing and Optimization", Ph.D. Dissertation, University of Tennessee at Chattanooga, August 2008.
4. Karman. S. L. Jr., and Sahasrabudhe, M., "Unstructured Elliptic Smoothing Revisited", AIAA-2009-1362, Orlando FL, January 2009.
5. http://people.nas.nasa.gov/~aftosmis/cart3d/
6. ACAD is a proprietary CAD program developed by Lockheed Martin Corporation.
7. http://www.pointwise.com
8. Hunton, Lynn W., Hicks, Raymond M., and Mendoza, Joel P., "Some Effects of Wing Planform on Sonic Boom", NASA TN D-7160, NASA, January 1973.
9. Karman, Steve L. Jr., Wooden, Perry, "CFD Modeling of F-35 Using Hybrid Unstructured Meshes", AIAA-2009-3662, San Antonio, TX, June 2009.
10. Karman, Steve L. Jr., "Unstructured Viscous Layer Insertion Using Linear Elastic Smoothing", AIAA Journal Vol. 45, Number 1, pp. 168-180, January 2007.